Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Urban Fatur

Mobilna aplikacija za ogled razstave

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Mira Trebar

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Mobilna aplikacija za ogled razstave v galeriji

Tematika naloge:

Ogled razstave je v galeriji običajno povezan s tekstovnimi opisi, ki so obešeni na steni ob slikah ali drugih eksponatih v prostoru. Kandidat naj zasnuje rešitev za implementacijo mobilne aplikacije, ki bo obiskovalca vodila po sobah v galeriji in mu omogočala ogled razstave in predstavitev informacij na mobilni napravi. Za predstavitev prototipnega sistema naj izdela spletno aplikacijo, ki omogoča izdelavo testne baze za Narodno galerijo v Ljubljani, kjer naj bo za identifikacijo razstave in prostorov uporabljena koda QR.

Zahvaljujem se družini za podporo pri študiju.
Posebna zahvala gre mentorici doc. dr. Miri Trebar za vso pomoč, nasvete in veliko mero potrpežljivosti pri izdelavi diplomskega dela.
Zahvaljujem se tudi vsem, ki so mi kakor koli pomagali pri študijskih obveznostih in polepšali dneve samega študija.
Hvala!

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ADT	Android Development Tools	Razvojna orodja za Android
API	Application Programming Interface	Programski vmesnik aplikacije
BLE	Bluetooth Low Energy	Nizkoenergetski Bluetooth
ID	Identification	Identifikacija
JSON	JavaScript Object Notation	JavaScript objekt za izmenjavo podatkov
JVM	Java Virtual Machine	Javanski navidezni stroj
MD5	Message digest	Kriptografska zgoščevalna funkcija
NFC	Near Field Communication	Komunikacija kratkega dosega
SQL	Structured Query Language	Strukturirani povpraševalni jezik za delo s podatkovnimi bazami
QR	Quick Response	Dvodimenzionalna črtna koda
REST	Representational State Transfer	Predstavitveni prenos stanja

Povzetek

Naslov: Mobilna aplikacija za ogled razstave

Avtor: Urban Fatur

V galerijah so opisi in informacije o razstavljenih eksponatih napisani na listih in nameščeni ob njih na steni.. Cilj diplomskega dela je bil razvoj rešitve, ki obiskovalcem galerije omogoča pridobivanje informacij o razstavi na pametnem telefonu. Spletna aplikacija je namenjena zaposlenim v galeriji. Omogoča dodajanje novih zapisov v sistem ter možnost pregleda in urejanja že obstoječih. Komunikacija med spletno aplikacijo in odjemalcem poteka preko storitve REST. Mobilna aplikacija se izvaja na operacijskem sistemu Android in je namenjena obiskovalcem galerije. Vstop v aplikacijo je zaščiten z vstopnim geslom razstave, ki si jo želi obiskovalec ogledati. S skeniranjem QR-kod, ki so postavljene ob vhodih v prostore se sprehaja po galeriji in dobiva informacije o eksponatih v slikovni, tekstovni in zvočni obliki.

Ključne besede: galerija, razstava, Android, mobilna aplikacija, QR-koda.

Abstract

Title: Mobile application to view the gallery exhibition

Author: Urban Fatur

In galleries, the information about exhibit pieces is written on sheets of paper and placed on their sides. The aim of this diploma thesis was to develop a solution, which would enable the visitors of the gallery, to obtain information about the exhibition on a smart phone. Web application is intended for the gallery employees. It enables to add new records into the system and offers the possibility to check and arrange the already existing ones. The communication between the web application and the user is carried out through the REST service. Mobile application is performed on Android operating system and it is meant to be used by visitors. The access to the application is protected with the password of the exhibition the visitor wishes to see. The visitor strolles through the exhibition with scanning the QR-codes, positioned at room entrances, and receives information about exhibits in visual, text and voice form.

Keywords: gallery, exhibition, Android, mobile application, QR-code.

Poglavje 1

Uvod

V številnih mestih po svetu obstajajo galerije, kjer so obiskovalcem predstavljeni zgodovinski dogodki in dela pomembnih umetnikov. Njihovi dosežki so urejeni in združeni na razstavah, ki so na vpogled obiskovalcem. Eksponati so v galerijah razstavljeni v večjem številu sob, ki so razporejene v več nadstropjih. Zaradi omejenega prostora in želje, da bi bilo obiskovalcem na voljo za ogled čim več eksponatov, prihaja do prostorske stiske. Ob odprtju novih razstav se v galerije napotijo množice ljudi. Ker je na stenah razstavljenih veliko slik in po prostoru postavljenih veliko kipov je pogosto pomembno, da se obiskovalci za nemoten ogled ne zadržujejo preveč časa pri posameznem eksponatu. Opis in informacije o posameznem eksponatu so napisane na listu in postavljene ob njem. Ker želijo obiskovalci dobiti čim več informacij, se pogosto dalj časa zadržujejo pred njim, kar posledično privede do gneče v galerijah.

Kljub razvoju in razširjeni uporabi pametnih telefonov lahko naštejemo veliko število galerij v Sloveniji, ki še nimajo mobilnih aplikacij, s katerimi bi obiskovalcem omogočale pridobivanje informacij o eksponatih, ki jih zanimajo. Mnoge večje svetovne galerije imajo svoje mobilne aplikacije in tako ponudijo obiskovalcem dodatne informacije o razstavljenih eksponatih.

Cilj diplomske naloge je bila implementacija mobilne aplikacije, ki jo bo obiskovalec uporabljal med obiskom galerije. V tem primeru bi lahko

zmanjšali gnečo pri ogledu eksponatov in ponudili obiskovalcem več informacij o razstavljenih eksponatih. Celotna naloga je sestavljena iz dveh aplikacij – spletne in mobilne. Spletna aplikacija je namenjena zaposlenim v galeriji. Zasnovana je tako, da omogoča dodajanje in urejanje podatkov tako o prostorih v galeriji kot tudi o vseh razstavah. Mobilna aplikacija je namenjena obiskovalcem. Z njo se pri ogledu razstav premikajo po prostorih galerije in pridobivajo podatke o eksponatih, ki jih zanimajo, v slikovni, tekstovni in zvočni obliki.

Poglavje 2

Tehnologije in programska oprema

Za izdelavo spletnih in mobilnih aplikacij je na voljo veliko tehnologij, orodij in programskih jezikov. Orodja so večinoma aplikacije, ki vsebujejo grafični uporabniški vmesnik, namenjen lažjemu delu uporabnika. Tehnologije sestavlja več smeri, ki so pomembne za predstavitev in delo s podatki, dinamičnim prikazovanjem ter komunikacijo. Programski jeziki so umetni jeziki, ki se uporabljajo za programiranje. Podrobno bomo opisali najpomembnejše tehnologije in programsko opremo, ki smo jo uporabili pri izdelavi diplomske naloge.

2.1 Mobilne tehnologije

Android

Android je programska platforma in operacijski sistem za pametne naprave, ki temelji na Linuxovem jedru [2]. Razvit je bil v podjetju Google in Open-Handset Alliance (OHA), ki predstavljata združenje več kot tridesetih podjetij s področij strojne in programske opreme, ter področja komunikacij [9]. Sprva je bil zasnovan predvsem za naprave na dotik, kot so mobilni telefoni in tablice. Uporabniška izkušnja temelji na neposredni manipulaciji uporabnika

z napravo. Naprava se odziva na uporabnikov dotik, pritisk in poteg. Zaradi vsestranskosti in izredne priljubljenosti se je Android razširil na televizijske zaslone, avtomobile in ročne ure. Prilagojene različice se uporabljajo tudi v prenosnih računalnikih, igralnih konzolah, digitalnih kamerah in drugih elektronskih napravah. Je vodilni operacijski sistem, namenjen pametnim telefonom. V prvem četrtletju 2017 je obsegal 85-odstotni delež vseh telefonov. Sledi mu iOS podjetja Apple s 14.7-odstotnim deležem [10].

QR-koda

QR-koda (angl. Quick Response) je matrična oz. dvodimenzionalna črtna koda. Leta 1994 jo je razvilo podjetje Denso Wave. Osnovni namen je bil ustvariti simbol, ki bo zlahka zaznan z napravo, kot je optični čitalnik [19]. QR-koda vsebuje informacije v dveh smereh (vertikalno in horizontalno), medtem ko osnovna črtna koda vsebuje podatke le v eni smeri. Zaradi tega lahko vsebuje večjo količino informacij kakor osnovna črtna koda. QR-koda ima v primerjavi z osnovno črtno kodo veliko prednosti:

- Medtem, ko osnovna črtna koda vsebuje največ 20 števil, lahko QR-koda vsebuje do stokrat več števil.
- Je desetkrat manjša od črtne kode z isto količino podatkov.
- Mogoče jo je prebrati, tudi če je 30% kode uničene.
- Ima možnost branja iz večjega števila kotov in strani.

Knjižnica ZXing

ZXing (angl. Zebra Crossing) je odprtokodna knjižnica, ki za svoje delovanje uporablja aplikacijo Barcode Scanner. Namenjena je skeniranju enodimenzionalnih in dvodimenzionalnih črtnih kod. Napisana je v programskem jeziku Java [3].

Knjižnica omogoča mobilni napravi branje podatkov, ki so zajeti s kamero na sami napravi. Za zajem slike mora aplikacija dostopati do kamere preko

namena (angl. Intent). Knjižnica zajeto sliko analizira in razbere črtno kodo ter vrne rezultat.

Barcode Scanner, ki ga uporablja knjižnica ZXing, je ena od najbolj priljubljenih aplikacij na portalu Google Play. Od maja 2016 naprej jo je na svoje mobilne telefone namestilo že več kot 126 milijonov uporabnikov[1].

2.2 Spletne tehnologije

Spletna storitev REST

REST (REpresentational State Transfer) je arhitekturni stil in pristop h komunikaciji med različnimi napravami preko interneta. Pogosto je uporabljen v povezavi s spletnimi storitvami [7].

Za komunikacijo se uporablja protokol HTTP (HyperText Transfer Protocol). REST uporablja osnovne HTTP–metode GET, POST, PUT in DELETE [12].

- GET: pridobivanje podatkov s strežnika.
- POST: pošiljanje podatkov na strežnik. Večinoma se uporablja za ustvarjanje novih vnosov, lahko pa tudi za posodabljanje že obstoječih.
- PUT: posodabljanje vnosov. PUT zamenja vse podatke obstoječega vnosa z novimi. Če v metodi PUT niso podani vsi elementi, ki jih vsebuje obstoječi vnos, se tem spremeni vrednost v empty ali null.
- DELETE: brisanje vnosov.

Odjemalec preko protokola HTTP pošlje zahtevo na dogovorjen URI (Uniform Resource Identifier), na katero mu strežnik odgovori. Odgovor je sestavljen iz statusne kode ter glave in podatkov HTTP. Statusna koda je trimestna koda, ki predstavlja informacije o odgovoru. Razdeljena je na pet vrst, vsaka vrsta ima svojo predpono. Statusne kode odgovora HTTP:

[1] https://play.google.com/store/apps/details?id=com.google.zxing.client.android

- 100 – informacija,
- 200 – uspeh,
- 300 – preusmeritev,
- 400 – napaka zahteve,
- 500 – napaka na strežniku.

Formata vrnjenih podatkov s strežnika sta lahko JSON ali XML.

JSON

JSON (JavaScript Object Notation) je preprost format za izmenjavo podatkov, neodvisen od programskega jezika. Je preprost in zaradi tega zelo priljubljen za uporabo. Podpirajo ga vsi moderni programski jeziki: C, C++, C#, Java, JavaScript, Perl, Python in mnogi drugi [17].

Tipi podatkov, ki se lahko prenašajo preko formata JSON, so niz, število, logična vrednost in null. Podatki se lahko prenašajo tudi v obliki struktur. Možni sta objekt in seznam [1]. Podatki v objektih so shranjeni v obliki ključ/vrednost. Do vsake vrednosti je omogočen dostop preko sklica na ustrezen ključ. Do podatkov v obliki seznama je dostop podan z indeksom, s kazalcem. JSON omogoča tudi kombinacijo teh dveh tipov.

XML

XML (Extensible Markup Language) je razširljiv označevalni jezik uporabljen za opis podatkov [8]. Uporablja se za izmenjavo podatkov na spletu in v večjih omrežjih. Po strukturi je zelo podoben jeziku HTML. Opis informacij je sestavljen iz elementov, definiranih z značkami. Element ima začetno in končno značko, njegova vrednost pa je zapisana med obema značkama. Velika prednost jezika XML je, da si lahko poljubno izberemo imena značk.

2.3 Orodja in programski jeziki

Android Studio

Android Studio je uradno razvojno okolje aplikacij Android. Temelji na razvojnem okolju IntelliJ IDEA. Program vsebuje grafični uporabniški vmesnik, ki programerjem olajša razvoj aplikacij.

Android SDK (Software Development Kit) je del Android Studia. Uporablja se za razvoj in prevajanje aplikacij Android. Android SDK je mogoče uporabljati s pomočjo konzole ali kot del katerega drugega razvojnega okolja (npr. Eclipse). Vsebuje knjižnice, ki jih aplikacije potrebujejo za delovanje, razhroščevalnik, emulator za testiranje aplikacij, dokumentacijo o programskih vmesnikih Android ter vzorčno kodo in vzorce primerov osnovne uporabe [13, 20]. Programer ima možnost razhroščevanja aplikacije na virtualni napravi ali na dejanski napravi preko USB-kabla. Pri programiranju smo uporabili tudi druge knjižnice (Picasso, PhotoViewAttacher, HttpURLConnection, JSONArray, ConnectivityManager). Večinoma so bile vključene za implementacijo posameznih delov kode, nekatere knjižnice pa za videz same aplikacije.

Visual Studio

Za integrirano razvojno okolje (IDE – Integrated Development Environment) smo izbrali Visual Studio. Spletno aplikacijo smo razvili v razvojnem okolju .NET s pomočjo programa Visual Studio 2015. Napisana je v programskem jeziku C#. Visual Studio je pogosta izbira programerjev, ki se odločijo za izdelavo programa v okolje .NET. Urejevalnik je pregleden, program pa ponuja samodejno dokončevanje kode. Uporabnik ima tudi možnost razhroščevanja kode in samodejne sinhronizacije za shranjevanje kode s portaloma GitHub in Microsoft Azure.

C#

C# (C Sharp) je objektno orientiran programski jezik. Razvil ga je Microsoft v okviru ogrodja .NET. Zgleduje se po številnih programskih jezikih – najbolj izrazito po C, C++ in Javi. Jezik je bil zelo skrbno načrtovan z namenom uporabiti izkušnje pri razvoju drugih programskih jezikov, uporabiti najboljše značilnosti in popraviti nekatere pomanjkljivosti, ki so se pokazale v razvoju teh jezikov [6]. Programi, napisani v C#, imajo zelo dobro prenosljivost med različnimi platformami.

Java

Java je visoko nivojski, objektno usmerjeni programski jezik, razvit v podjetju Sun Microsystems. Deluje na različnih operacijskih sistemih, kot so Windows, Mac OS in različne verzije sistema UNIX [16]. Različica 1.0 je bila objavljena leta 1995 [14] in se redno posodablja. Trenutno je najnovejša delujoča verzija Java SE 8.

Glavna prednost Jave je besedna zveza "Write Once, Run Anywhere". Pomeni, da se program, napisan v tem programskem jeziku, lahko izvaja na vsaki napravi, ki ima nameščen JVM (navidezni računalnik, ki ga lahko implementiramo na različnih resničnih računalnikih). Java ima na voljo veliko knjižnic, ki programerjem omogočajo različne dodatne možnosti pri samem programiranju.

SQL

Širitev dinamičnih spletnih strani je v veliki meri glavni razlog, da je večina podatkov shranjenih v podatkovnih bazah. Upravljanje s podatkovnimi bazami je zapleten postopek, zaradi katerega je bil razvit jezik SQL (Structured Query Language). Že samo ime pove, da se uporablja za poizvedovanje in delo s podatki [5].

Ne glede na to, da lahko podatke v tabelah sortiramo, filtriramo in iščemo po njih, moramo pogostokrat iz obsežnih tabel dobiti le del podatkov. Za

to se uporabljajo poizvedbe. Najbolj osnovna poizvedba se začne z ukazom SELECT. Rezultati poizvedbe so podatki ene ali več tabel. Za spreminjanje podatkov, ki so shranjeni v tabelah, uporabljamo poizvedbe, ki se začnejo z ukazi INSERT, UPDATE ali DELETE.

Poglavje 3

Načrtovanje

V fazi načrtovanja smo definirali osnovne zahteve za predstavitev razstave, ki bi jo lahko obiskovalec galerije imel na svojem pametnem telefonu. Dogovorili smo se, da bomo obiskovalcu prikazali podatke v slikovni, tekstovni in audio obliki. Poiskati smo morali podatke o trenutnih razstavah v galeriji in razstavljenih eksponatih.

Cilj diplomske naloge je bila izdelava mobilne aplikacije za ogled galerije, ki bi omogočala uporabnikom pridobivanje informacij o eksponatih. Za dostop mobilne naprave do želenih informacij smo morali izdelali še spletno aplikacijo, katere glavni namen je dodajanje podatkov in datotek ter njihovo urejanje v podatkovni bazi. Za uporaben in zanimiv primer razstave pri zasnovi aplikacij smo se dogovorili, da bo rešitev zasnovana za Narodno galerijo. Odločitev je temeljila predvsem na dejstvu, da je Narodna galerija največja galerija v Sloveniji in ima razstavljenih veliko eksponatov. Prav tako nima podobne mobilne aplikacije, namenjene obiskovalcem. Na spletu je objavljena aplikacija z naslovom Virtualni vodič po stalni zbirki Slovenska umetnost [18], kar predstavlja vizualni sprehod po galeriji. Obiskovalci vidijo le prostore v galeriji, ne dobijo pa informacij o razstavljenih eksponatih.

Narodna galerija Slovenije je največja galerija slovenskih likovnih del. Je v Ljubljani. Ustanovljena je bila leta 1918. Stavbo so večkrat prenavljali in sedanjo podobo je dobila leta 2016. Takrat so se enotno povezali vsi

trije stavbni deli galerije [4]. Za obiskovalce je vedno odprta stalna razstava, ki obsega predvsem dela slovenskih umetnikov. Poleg stalne razstave se v galeriji izmenjujejo še različne druge razstave.

Osnovne funkcionalnosti mobilne aplikacije smo povzeli po aplikacijah, ki so na voljo obiskovalcem za ogled galerij ali muzejev (Natural History Museum[1], London National Gallery[2], Regionalni muzej Koper[3] in muzej Rodin[4]). Vizualno podobo aplikacije smo zasnovali in izdelali tako, da smo za vsako okno spletne ter mobilne aplikacije izdelali načrt strani (angl. wireframe). Z njim smo določili osnovno postavitev elementov na strani. Podrobnosti in ostale vizualne značilnosti smo izbrali med samo implementacijo rešitve.

3.1 Spletna aplikacija

Spletna aplikacija je namenjena zaposlenim v galeriji. Omogoča dodajanje novih elementov v podatkovno bazo, urejanje, pregled in spreminjanje atributov že obstoječih elementov.

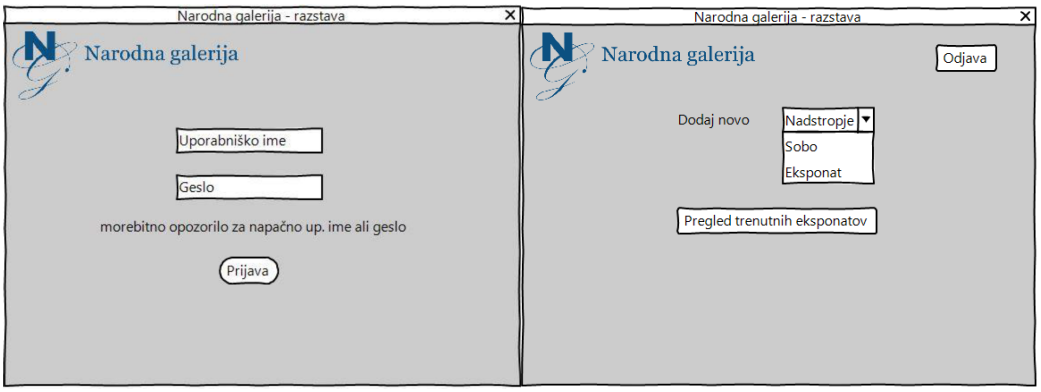
3.1.1 Stran za prijavo

Slika 3.1a prikazuje osnovno stran spletne aplikacije za prijavo. V levem zgornjem kotu je logotip Narodne galerije. V sredini okna sta polji za uporabniško ime in geslo uporabnika ter gumb Prijava. Ob napačni prijavi je uporabnik obveščen z opozorilom nad gumbom za prijavo. Med samo implementacijo rešitve smo se odločili, da bomo na prvo stran dodali še polje za registracijo novega uporabnika.

[1] https://play.google.com/store/apps/details?id=air.com.nhm.london.vusiem
[2] https://play.google.com/store/apps/details?id=com.macsoftex.amazongallery
[3] https://play.google.com/store/apps/details?id=si.ngn.pmk2
[4] https://play.google.com/store/apps/details?id=air.com.rodin.paris.vusiem

3.1.2 Prva stran

Slika 3.1b prikazuje prvo stran po prijavi v spletno aplikacijo. Uporabnik ima za dodajanje novih elementov v sistem na voljo spustni meni z možnostmi za dodajanje novega nadstropja, sobe in eksponata. Pod spustnim menijem je gumb, ki uporabnika pripelje na stran, kjer so prikazani vse eksponati, ki so trenutno v bazi. Uporabnik se lahko odjavi s klikom na gumb Odjava. Med implementacijo je bil spustni meni odstranjen. Dodali smo nove gumbe, vsakega za dodajanje novega elementa v sistem. Za ta korak smo se odločili predvsem zaradi lepšega videza, ter morebitnih težav uporabnikov pri delu s spustnim menijem. Dodali smo tudi gumb, ki se uporablja za dodajanje nove razstave v sistem.



(a) (b)

Slika 3.1: Načrt oken za: a) prijavno stran; b) prvo stran

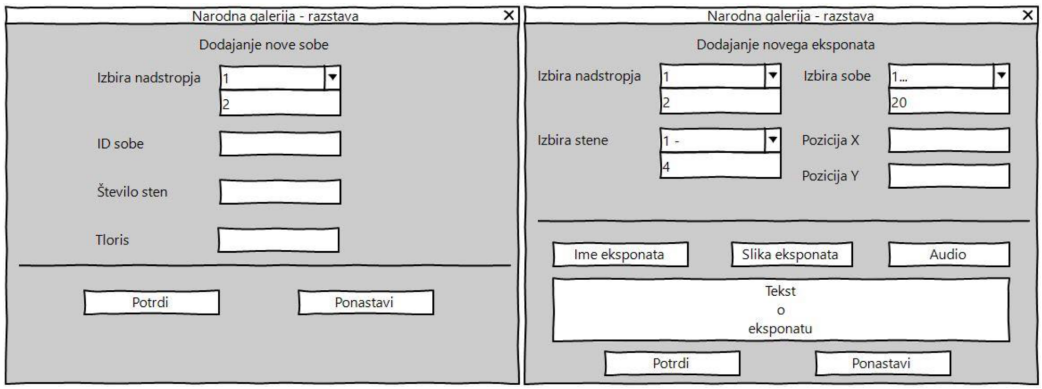
3.1.3 Dodajanje sobe

Slika 3.2a prikazuje okno za dodajanje nove sobe v sistem. Uporabnik ima na izbiro spustni seznam z nadstropji in izbere tistega, v katerem bo soba. Za vsako sobo mora izbrati ID–številko, število sten in tloris. Uporabniku sta na voljo gumba Potrdi in Ponastavi. Gumb Potrdi se uporablja za dodajanje nove sobe v sistem, medtem ko gumb Ponastavi izbriše podatke iz vnosnih polj. Med implementacijo se je okno spremenilo. Odstranili smo polje za

vpis ID sobe, saj se ta kreira sam ob vnosu v podatkovno bazo.

3.1.4 Dodajanje eksponata

Slika 3.2b prikazuje okno za dodajanje novega eksponata. Uporabnik mora izbrati nadstropje, sobo, steno in položaj eksponata. Vnesti mora še ime eksponata, sliko, zvok in tekst za opis eksponata. S klikom na gumb Potrdi doda nov eksponat v sistem. Klik na gumb Ponastavi izbriše podatke iz vnosnih polj. Med implementacijo se je postavitev okna spremenila. Dodali smo še spustni seznam za izbiro razstave ter vnosno polje o avtorju eksponata. Ker ima mobilna aplikacija možno izbiro večjezičnosti, smo morali dodati še polja za angleško ime eksponata, zvočno vsebino in tekst.

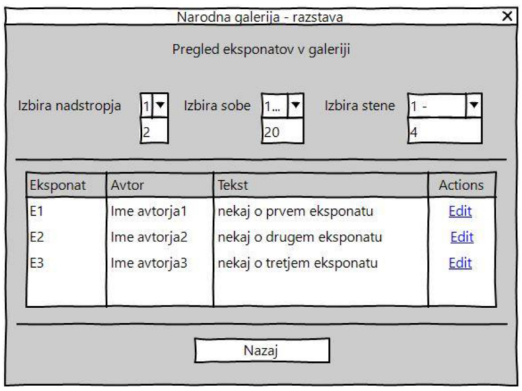


(a) (b)

Slika 3.2: Načrt oken za dodajanje: a) sobe; b) eksponata

3.1.5 Pregled eksponatov

Slika 3.3 predstavlja okno za pregled eksponatov v bazi. Prikazani so v tabeli, kjer ima uporabnik za vsak eksponat možnost popravljanja ali brisanja. Nad tabelo so trije spustni meniji namenjeni sortiranju eksponatov pred prikazom v tabeli. Končna rešitev vsebuje le spustni meni za izbiro razstave, saj smo v tabeli omogočili možnost sortiranja po atributih.



Slika 3.3: Načrt okna za pregled vnešenih eksponatov

3.2 Mobilna aplikacija

Mobilna aplikacija bo uporabniku omogočala pridobivanje informacij o eksponatih za eno ali več razstav v galeriji. Informacije o posameznem eksponatu so na voljo v slikovni, tekstovni in zvočni obliki. Uporabnik ima možnost pregleda nadstropij v galeriji. Med sobami se lahko premika z vpisom oznake sobe ali s skeniranjem QR-kode pri vhodu v sobo.

3.2.1 Vstopna stran

Slika 3.4a prikazuje vstopno stran mobilne aplikacije, ki na zaslonu predstavlja sliko vhoda v Narodno galerijo. V levem spodnjem kotu je povezava na uradno stran galerije na družabnem omrežju Facebook, desno od nje pa povezava na uradno spletno stran galerije. V desnem kotu je gumb za menjavo jezika, ki pa je bil kasneje odstranjen. Odločili smo se, da je bolj primerno, da je aplikacija na voljo v jeziku, ki ga ima uporabnik izbranega na pametnem telefonu. Uporabnik se lahko prijavi z vpisom vstopne kode ali pa skeniranjem QR-kode, ki je natisnjena na vstopnici.

3.2.2 Prva stran

Slika 3.4b prikazuje stran z nadstropji Narodne galerije. Uporabnik se med nadstropji pomika z gumboma gor in dol, ki sta bila implementirana v izdelavi aplikacije. Vse sobe na slikah so označene s številkami – njihovimi ID-ji. Uporabnik se lahko premakne v sobo z vpisom ID-ja sobe ali pa skeniranja QR-kode, ki je pri vsakem vhodu v prostor.

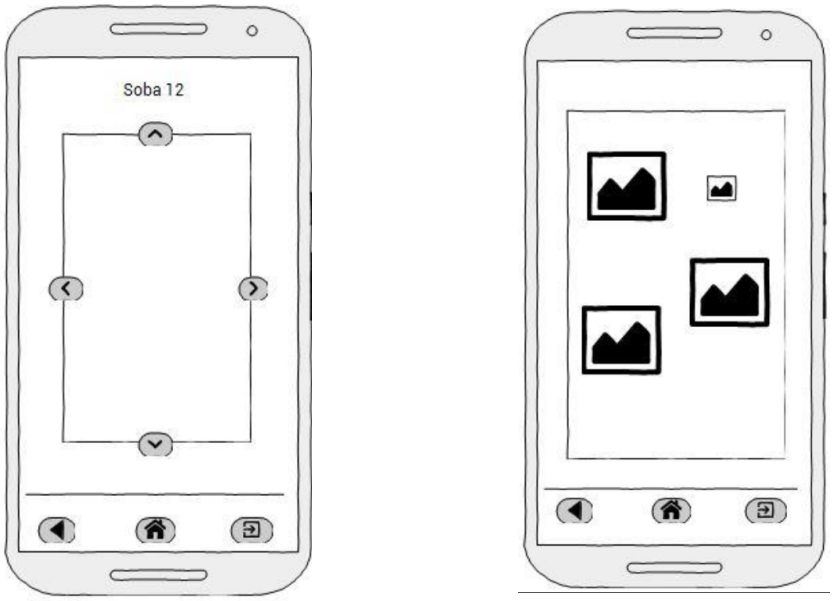
(a) (b)

Slika 3.4: Načrt oken za: a) vstopno stran; b) pregled nadstropij

3.2.3 Izbira sobe in stene

Slika 3.5a prikazuje načrt okna za pregled sobe. Večji del je predstavljen s tlorisom sobe. Gumbi, ki so poleg tlorisa, predstavljajo določeno steno. S klikom na gumb poleg stene se prikaže seznam eksponatov, ki so razstavljeni na izbrani steni. Med izdelavo smo pod tlorisom dodali še gumb Prikaži kipe in s tem uporabniku omogočili pregled eksponatov, razstavljenih po prostoru. Slika 3.5b predstavlja posamezno steno. Na njej so slike eksponatov, ki so na voljo v sistemu. Eksponati so po steni postavljeni glede na vneseni

koordinati X in Y v spletni aplikaciji. Med izdelavo smo prikaz eksponatov na steni zamenjali z eksponati, prikazanimi v seznamu..

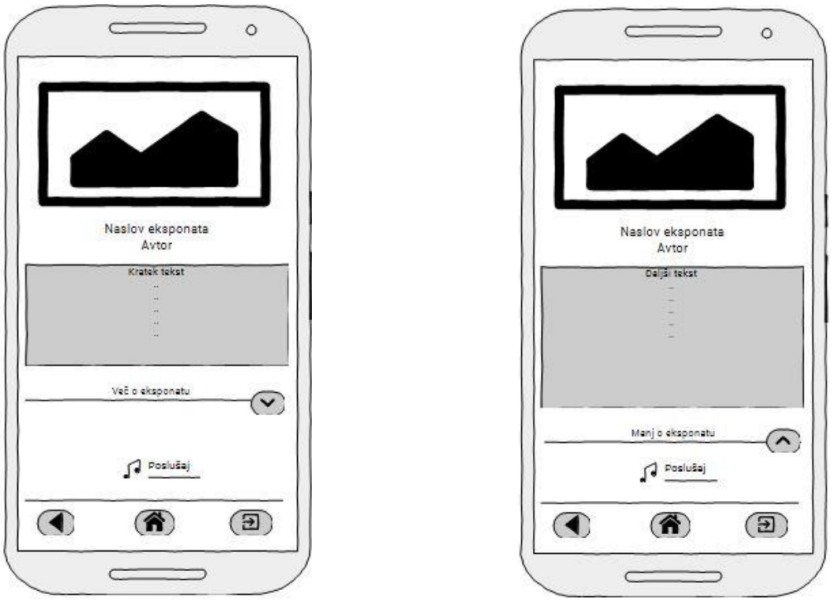


(a) (b)

Slika 3.5: Načrt oken za pregled: a) sobe; b) stene

3.2.4 Prikaz informacij o eksponatu

Slika 3.6 prikazuje načrt oken za pregled eksponata. V zgornjem delu okna je slika eksponata, ki ji sledita naslov in podatki o avtorju. Večji del strani zapolnjuje tekst. Na sliki 3.6a je prikazan krajši tekst. Ob kliku na gumb Več o eksponatu se bo tekst razširil in tako uporabniku ponudil več informacij (slika 3.6b). To možnost smo kasneje spremenili v premično tekstovno polje. V spodnjem delu okna je gumb za poslušanje zvočne vsebine.



(a) (b)

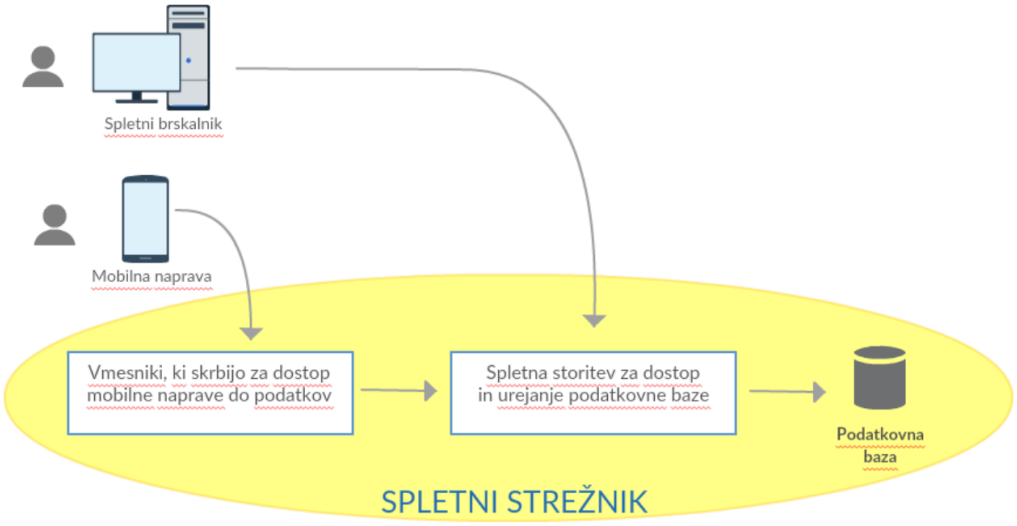
Slika 3.6: Načrt okna za pregled eksponata: a) poglej več; b) poglej manj

3.3 Arhitektura sistema

Aplikacija za ogled razstave je sestavljena iz dveh delov. Prvi strežniški del je napisan v ogrodju .NET. HTTP-strežnik je verzije IIS 10 podjetja Microsoft. Strežniška aplikacija je objavljena na spletnem ponudniku Somee [11], ki je eden redkih brezplačnih spletnih ponudnikov, ki omogočajo gostovanja spletnih strani .NET. Ponudnik Somee omogoča gostovanje, objavo podatkovne baze in prostor za shranjevanje datotek (angl. file storage). Strežniški del je sestavljen iz podatkovne baze, spletne strani, ki je namenjena zaposlenim v Narodni galeriji in množice API-jev, preko katerih strežnik ponuja svoje podatke in storitve različnim odjemalcem. Mobilna aplikacija je implementirana v operacijskem sistemu Android. Testirana je bila na mobilnem telefonu LG K10 z verzijo operacijskega sistema Android 6.0.

Mobilna aplikacija ima vlogo odjemalca. Vse podatke, ki so potrebni za delovanje, pridobi iz strežnika. Zaradi optimalnega delovanja aplikacije mora

imeti mobilna naprava vzpostavljeno internetno povezavo za komunikacijo s spletnim strežnikom.. Slika 3.7 prikazuje arhitekturo programske rešitve.



Slika 3.7: Arhitektura sistema za ogled razstave na mobilni napravi

3.4 Podatkovni model

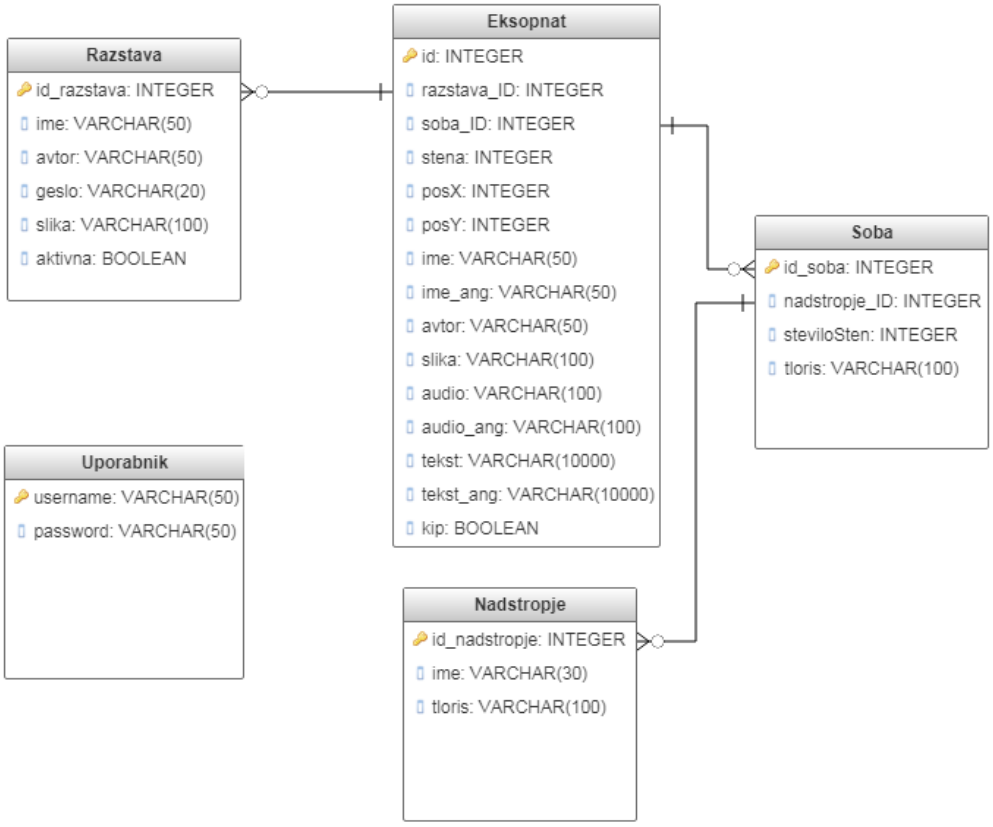
Osnovno obliko podatkovnega modela smo si zamislili že pred začetkom implementacije programske rešitve. Med razvojem se je model začel spreminjati, saj smo dobivali nove in boljše ideje ter aplikacijama dodajali dodatne funkcionalnosti. Ob zaključku diplomske naloge je podatkovni model sestavljen iz petih tabel, prikazanih na sliki 3.8.

Uporabnik

V tabeli Uporabnik so shranjeni podatki o registriranih uporabnikih spletne aplikacije. Tabela ni v povezavi z nobeno izmed ostalih štirih tabel.

- **username** – uporabniško ime uporabnika.

- **password** – geslo uporabnika shranjeno v zgoščeni obliki. Za zgoščevanje je bil uporabljen algoritem MD5 [15].



Slika 3.8: Podatkovni model za urejanja razstav

Razstava

Tabela Razstava vsebuje podatke o razstavah, ki so na voljo za ogled. Vsi atributi razen atributa *avtor* so obvezni pri vnosu nove razstave.

- **id_razstava** – enolična oznaka (ID) posamezne razstave.
- **ime** – ime, ki bo predstavljalo določeno razstavo.

- **avtor** – avtor ali več avtorjev razstave, atribut je lahko enak null, če razstava nima podatka o avtorju.
- **geslo** – vstopno geslo za mobilno aplikacijo. Atribut geslo je koda, ki jo vnese ali skenira uporabnik. Polje je obvezno, saj v nasprotnem primeru ne moremo dostopati do eksponatov razstave v mobilni aplikaciji.
- **slika** – URL do slike razstave. Slika je uporabljena na drugi strani mobilne aplikacije.
- **aktivna** – logična vrednost 0 ali 1. Ob vnosu nove razstave se vrednost avtomatsko nastavi na 1, kar pomeni, da je ogled razstave trenutno mogoč. Logična vrednost 0 pomeni, da razstava ni več na voljo. Uporabnik spletne aplikacije lahko spreminja ta atribut.

Nadstropje

Tabela Nadstropje vsebuje informacije o nadstropjih. Vsako nadstropje je predstavljeno z imenom in sliko tlorisa. Vsi atributi so obvezni ob vnosu novega nadstropja.

- **id_nadstropje** – enolična oznaka (ID) posameznega nadstropja.
- **ime** – ime nadstropja, kakor bo shranjeno v sistemu.
- **tloris** – URL do tlorisa nadstropja.

Soba

Tabela Soba vsebuje informacije o sobah. Posamezno sobo opisujejo ID, ID nadstropja, število sten in tloris sobe. Vsi atributi so obvezni ob vnosu nove sobe.

- **id_soba** – enolična oznaka (ID) posamezne sobe.
- **nadstropje_ID** – referenca na tabelo Nadstropje. Za vsako dodano sobo je določeno, v katerem nadstropju je.

- **steviloSten** – število sten, ki jih ima soba.
- **tloris** – URL do tlorisa sobe.

Eksponat

Tabela Eksponat vsebuje informacije o eksponatih, ki so vneseni v sistem. Večje število atributov je potrebno predvsem zaradi referenc do ostalih tabel in podpore večjezičnosti.

- **id** – enolična oznaka (ID) posameznega eksponata.
- **razstava_ID** – referenca na tabelo Razstava. Za vsak dodan eksponat je določeno, v katero razstavo spada.
- **soba_ID** – referenca na tabelo Soba. Za vsak dodan eksponat je določeno v kateri sobi bo.
- **stena** – predstavlja steno v določeni sobi, na kateri bo eksponat razstavljen.
- **posX** – položaj (koordinata X) eksponata na steni. Je pozitivno celo število, eksponat na levi strani stene ima manjšo koordinato X kot eksponat, ki je razstavljen na desni strani stene. Atribut se uporablja pri sortiranju eksponatov v mobilni aplikaciji.
- **posY** – Y-koordinata eksponata na steni. Je pozitivno celo število, eksponat na zgornji polovici stene ima manjšo koordinato Y kot eksponat, razstavljen na spodnji polovici.
- **ime** – ime eksponata, ki bo vključeno v bazi.
- **ime_ang** – angleško ime eksponata.
- **avtor** – avtor eksponata, atribut ni obvezen pri dodajanju novega eksponata v sistem.

- **slika** – URL do slike eksponata. Slika se uporablja v mobilni aplikaciji pri pregledu posameznega eksponata.
- **audio** – URL do zvočne datoteke v slovenskem jeziku za posamezen eksponat. Atribut ni obvezen ob dodajanju novega eksponata v sistem.
- **audio_ang** – URL do zvočne datoteke v angleškem jeziku za posamezen eksponat. Atribut ni obvezen ob dodajanju novega eksponata v sistem.
- **tekst** – slovenski opis eksponata.
- **tekst_ang** – angleški opis eksponata.
- **kip** – logična vrednost 0 ali 1. Uporabnik jo izbere pri vnosu novega eksponata v spletni aplikaciji. Vrednost 0 pomeni, da je eksponat slika, ki je razstavljena na določeni steni. Vrednost 1 pomeni, da eksponat ni razstavljen na steni, ampak je nameščen nekje v sobi kot kip.

Poglavje 4

Implementacija

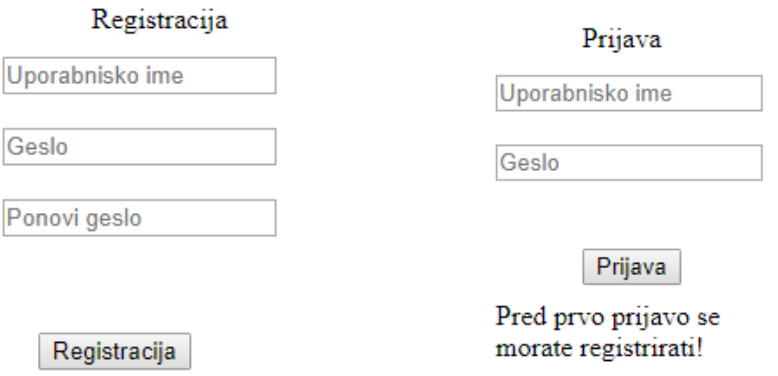
V diplomski nalogi sta bili izdelani in testirani spletna ter mobilna aplikacija. Predstavljene so vse možne funkcionalnosti in izseki kode, ki so pomembni za njuno delovanje.

4.1 Urejanje razstave

V nadaljevanju je za spletno aplikacijo predstavljeno, kako potekajo faze dodajanja novih elementov v sistem in spreminjanje atributov že obstoječih. Spletna aplikacija je bila vse do zaključka testiranja dostopna na naslovu http://www.asdf.somee.com.

4.1.1 Prijavno okno

Prijavno okno je začetna stran spletne aplikacije. Razdeljeno je na dva obrazca – obrazec za registracijo novega uporabnika in obrazec za prijavo že obstoječega uporabnika. Novi uporabniki lahko vnesejo svoje podatke v obrazec za registracijo uporabnika (slika 4.1a). Če je bil uporabniški račun uspešno ustvarjen, se uporabniku prikaže obvestilo. V spletno aplikacijo se lahko prijavijo osebe, ki že imajo ustvarjen uporabniški račun. Obstoječi uporabniki vnesejo svoje podatke v obrazec za prijavo (slika 4.1b) in kliknejo gumb Potrdi. Pri obeh obrazcih preverjamo vnose uporabnika. Pri obrazcu



(a) Obrazec za registracijo

(b) Obrazec za prijavo

Slika 4.1: Prijavno okno spletne aplikacije

za registracijo preverjamo, ali uporabniško ime že obstaja in ali se vpisani gesli ujemata. Vpisano geslo kriptiramo z algoritmom MD5. Pri obrazcu za prijavo primerjamo vnesene podatke s podatki uporabnika v podatkovni bazi. V primeru napake uporabniku prikažemo ustrezno obvestilo. V primeru uspešne prijave je uporabnik preusmerjen na prvo stran spletne aplikacije.

4.1.2 Prva stran

Prva stran spletne aplikacije (slika 4.2) je namenjena navigaciji med vsemi možnostmi, ki so na voljo. Zgornji del strani je statičen. Uporabnik ima možnost odjave s klikom na gumb Odjava. Osrednji štirje gumbi so namenjeni dodajanju novih elementov v podatkovno bazo. Spodnja gumba sta namenjena pregledu in spreminjanju atributov že vnesenih razstav in eksponatov.

V spodnjem delu strani je postavljen dinamični gradnik *iframe*. Primeren je za prikazovanje dinamičnih vsebin. Z gradnikom iframe na prvi strani prikazujemo okna za dodajanje nove razstave, nadstropja, sobe in eksponate. Pred prikazom posameznega okna moramo določiti atribute gradnika iframe. Primer spreminjanja atributov prikazuje slika 4.3. V tem gradniku ne pri-

Slika 4.2: Prva stran spletne aplikacije

kazujemo oken za pregled trenutnih eksponatov in razstav, ker sta preveliki. Zaradi boljše preglednosti ju prikažemo v svojem oknu.

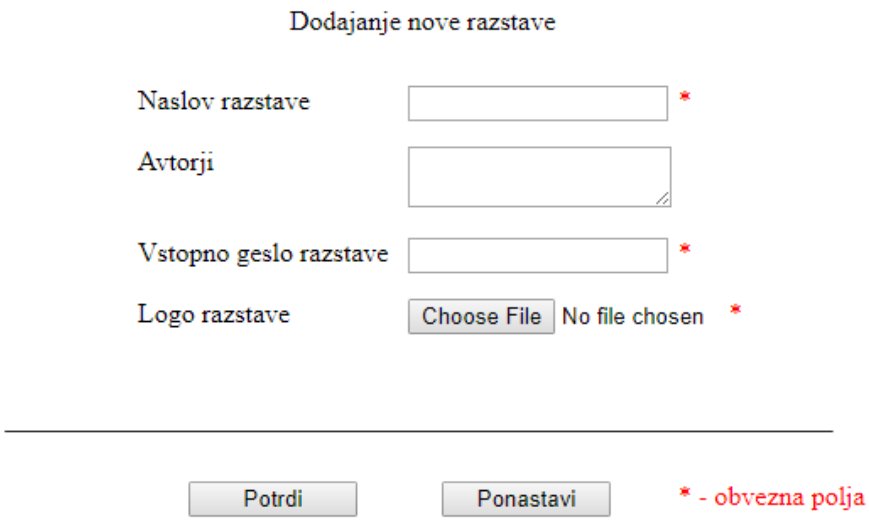
4.1.3 Dodajanje nove razstave

Slika 4.4 prikazuje vnosna polja za dodajanje nove razstave v podatkovno bazo. Pri vnosu moramo obvezno vnesti ime razstave. Tekstovno polje Avtorji ni obvezno, saj ni nujno, da bo imela vnesena razstava avtorja ali več morebitnih avtorjev. Vnesti moramo vstopno geslo razstave. Uporablja se

```
0 references | ufatur, 36 days ago | 1 author, 1 change
protected void ButtonSoba_Click(object sender, EventArgs e)
{
    EnableGumbe();
    ButtonSoba.Enabled = false;
    frameVsebina.Attributes["src"] = "Soba.aspx";
    frameVsebina.Attributes["width"] = "600px";
    frameVsebina.Attributes["height"] = "320px";
    frameVsebina.Visible = true;
}
```

Slika 4.3: Sprememba atributov gradnika iframe za prikaz okna za dodajanje nove sobe

kot prijavno geslo pri mobilni aplikaciji. Razstave imajo lahko ista vstopna gesla, če želimo, da si lahko obiskovalci galerije z istim vstopnim geslom ogledajo eksponate večjega števila razstav. Polje Logo razstave je obvezno. Izbrati moramo sliko, ki bo prikazovala razstavo na prvi strani mobilne aplikacije.



Slika 4.4: Stran za dodajanje nove razstave

Ob kliku na gumb Potrdi preverjamo vpisane in shranjene podatke na strani odjemalca (uporabljamo programski jezik JavaScript), kot tudi na strani strežnika. Pri vnosnih poljih za naslov in geslo razstave preverjamo dolžino vpisanega niza. Ob vnesenih neustreznih podatkih obvestimo uporabnika. Pri vnosnem polju za logo razstave preverjamo:

- **Ime datoteke:** Pozorni smo na končnico. Dovolili smo le datoteke tipa jpg in png (slika 4.5a). Pri imenu datoteke smo pozorni tudi na prisotnost šumnikov v imenu. Ob kršitvi katerega izmed pogojev uporabnika obvestimo in prekinemo dodajanje nove razstave.
- **Velikost datoteke:** Največja velikost, ki jo slikovna datoteka lahko doseže je 190 KB (slika 4.5b). Ob preveliki datoteki se uporabniku prikaže obvestilo.

```
<asp:RegularExpressionValidator ID="regexValidator" runat="server"
ForeColor="Red" style="left:240px; top:230px; position:absolute"
    ControlToValidate="FileUploadRazstava"
    ErrorMessage="Le JPG/PNG datoteke omogočene."
    ValidationExpression="(.*\.([JjPp][PpNn][Gg])|.*\.([Jj][Pp][Ee][Gg])$)">
</asp:RegularExpressionValidator>
```

(a) Preverjanje tipa, končnice datoteke

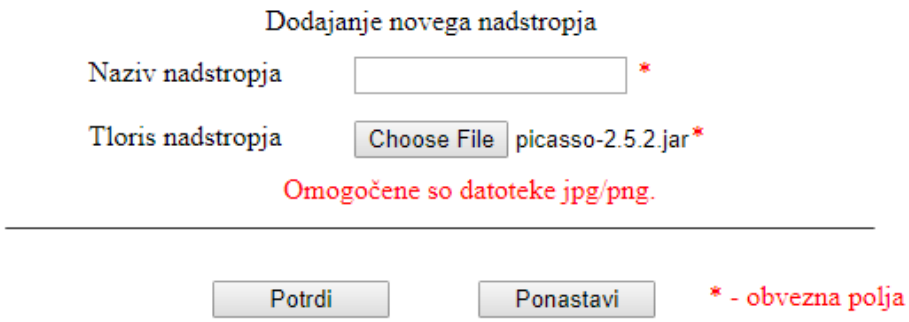
```
<script type="text/javascript">
    function Validate() {
        var uploadFile = document.getElementById("<%=FileUploadRazstava.ClientID%>");
        if (uploadFile.files[0].size >= 194560) {
            window.alert("Največja velikost datoteke je lahko 190KB!");
            return false;
        }
        if (uploadFile.files[0].size == 0) {
            window.alert("Vnesite logo razstave!");
            return false;
        }
        var file = uploadFile.files[0].name;
        var array = ['š', 'č', 'ž', 'Š', 'Č', 'Ž', 'Ć', 'ć', 'đ', 'Đ'];
        if (file.indexOf(array) >= 0) {
            window.alert("Ime datoteke ne sme vsebovati šumnikov!");
            return false;
        }
```

(b) Preverjanje imena in velikosti datoteke

Slika 4.5: Preverjanje vnosov pri dodajanju nove razstave

4.1.4 Dodajanje novega nadstropja

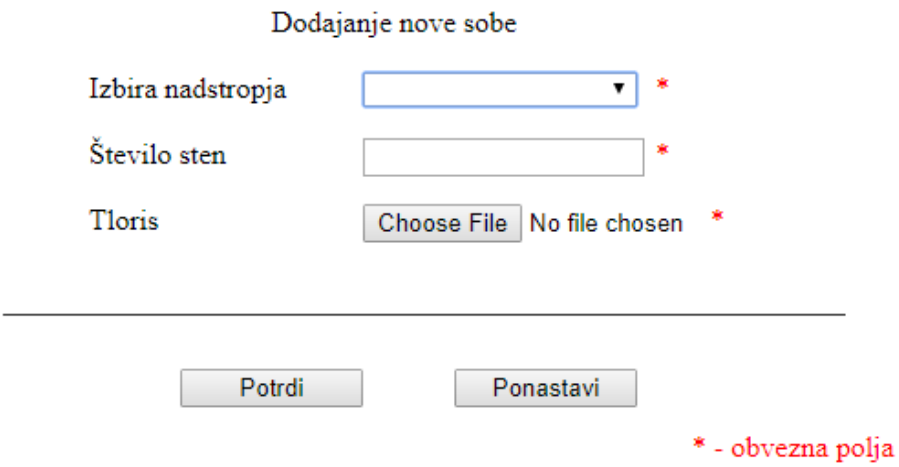
Okno je namenjeno dodajanju novega nadstropja v podatkovno bazo. Izbrati moramo ime nadstropja in sliko tlorisa nadstropja. Obe vnosni polji sta obvezni. Tudi tukaj preverjamo vnesene podatke. Koda je ista, kakor pri oknu za dodajanje nove razstave (sliki 4.5b in 4.5a). Ob neustrezno vnesenih podatkih uporabniku prikažemo obvestilo o napaki. Slika 4.6 prikazuje obvestilo, ki se uporabniku prikaže, če želi kot tloris nadstropja izbrati datoteko s končnico jar.



Slika 4.6: Prikaz obvestila o napačnem tipu datoteke

4.1.5 Dodajanje nove sobe

Novo sobo lahko v sistem dodamo z oknom, prikazanim na sliki 4.7. Vsa vnosna polja so obvezna. Pri dodajanju sobe moramo določiti nadstropje. V spustnem meniju imamo na voljo vsa dosedanja nadstropja, ki so vnesena v podatkovno bazo. Vnosno polje za določitev števila sten smo filtrirali. Trenutna rešitev je zasnovana za štiri stene in ne za dejansko število sten, kot jih ima posamezna soba. Omogočili smo le vpis števil. Polje preverjamo s pogojem, če je vnesena vrednost večja od 0. Preverjanje podatkov o tlorisu sobe izvaja koda, opisana v poglavju 4.1.3.



Slika 4.7: Okno za dodajanje nove sobe

4.1.6 Dodajanje novega eksponata

Slika 4.8 prikazuje okno za dodajanje novega eksponata. Ob prikazu okna so vsa vnosna polja zaklenjena. Uporabnik mora pred dodajanjem novega eksponata aplikaciji sporočiti, ali bo dodan eksponat slika ali kip. Glede na izbiro uporabnika se vnosna polja ustrezno odklenejo. V primeru, da želi vnesti sliko, se odprejo vsa vnosna polja. Če želi kip, se odprejo vsa vnosna polja razen spustnega menija za izbiro stene. Spustna menija za izbiro razstave in nadstropja sta statična. Ob vsakem nalaganju okna ju napolnimo s poizvedbo, ki preverja informacije o razstavah in nadstropjih, ki so vnesena v podatkovno bazo.

Spustna menija za izbiro sobe in stene, na kateri bo eksponat, sta dinamična. Spreminjata se glede na izbiro uporabnika v spustnem meniju za izbiro nadstropja. Slika 4.9 prikazuje kodo, ki dinamično dodaja podatke v omenjena spustna menija. Ko uporabnik izbere želeno nadstropje, shranimo njegovo odločitev v obliki seje. Izvede se poizvedba (angl. query), ki kot rezultat dobi ID-je sob in njihovo število sten v nadstropju, ki si ga je uporabnik izbral. Podatki o ID-jih sob se samodejno dodajo v spustni meni za izbiro sobe. Glede na izbiro se izvede nova poizvedba, katere rezultat je

Slika 4.8: Okno za dodajanje novega eksponata

število sten v sobi. Podatke vnesemo v spustni meni za izbiro stene. Ob spremembi indeksa pri spustnem meniju za izbiro sobe se poleg okna za dodajanje novega eksponata prikaže tloris sobe. Uporabniku je tako olajšana izbira pravilne stene. Dogovorili smo se, da bo stena 1 na spodnji strani tlorisa. Ostale stene ji sledijo v smeri urinega kazalca. Vnosni polji za izbiro koordinat X in Y sta pomembni za položaj eksponata na steni. Če uporabnik vnaša v sistem kip, pomeni ta položaj eksponata v prostoru. Uporabnik mora vnesti podatke še v ostala obvezna vnosna polja.

Vnesene podatke preverjamo na strani uporabnika in strežnika. Tekstovnim vnosnim poljem preverjamo dolžino vnesenega niza. Vnosni polji za dodajanje slik preverja koda na sliki 4.5. Vnosno polje za dodajanje zvočne vsebine preverja koda na sliki 4.10. Kot možne formate dovolimo le tipe datotek mp3. Velikost zvočne datoteke ne sme presegati 1 MB. V nasprotnem primeru aplikacija uporabnika obvesti, da želi naložiti preveliko zvočno datoteko.

```
if (Page.IsPostBack)
{
    if (Int32.TryParse(DropDownStena.SelectedValue, out temp)){
        temp = Int32.Parse(DropDownStena.SelectedValue);
    }
    DropDownStena.Items.Clear();
    if (DropDownNadstropje.SelectedIndex != (int)Session["CurrentNadID"]) {
        Session["CurrentNadID"] = DropDownNadstropje.SelectedIndex;

        using (SqlConnection sqlconnection = new SqlConnection(connectionStrings))
        {
            string selectQuery = "SELECT [id_soba], [steviloSten] from Soba WHERE [nadstropje_ID] = @ID";
            SqlCommand selectCommand = new SqlCommand(selectQuery, sqlconnection);
            selectCommand.Parameters.AddWithValue("@ID", DropDownNadstropje.SelectedValue);
            try
            {
                sqlconnection.Open();
                DropDownSoba.DataSource = selectCommand.ExecuteReader();
                DropDownSoba.DataTextField = "id_soba";
                DropDownSoba.DataBind();
                DropDownSoba.Items.Insert(0, new ListItem("", ""));

                if (DropDownSoba.SelectedValue != "" ) {
                    FetchImage(sender, e);
                    using (SqlConnection sqlconnection2 = new SqlConnection(connectionStrings))
                    {
                        string selectQuery2 = "SELECT [steviloSten] from Soba WHERE [id_soba] = @ID";
                        SqlCommand selectCommand2 = new SqlCommand(selectQuery2, sqlconnection2);
                        selectCommand2.Parameters.AddWithValue("@ID", DropDownSoba.SelectedValue);
                        try
                        {
                            sqlconnection2.Open();
                            if (Int32.TryParse(selectCommand2.ExecuteScalar().ToString(), out max))
                            {
                                DropDownStena.Items.Add(new ListItem("", ""));
                                for (int i = 1; i <= max; i++)
                                {
                                    DropDownStena.Items.Add(new ListItem(i.ToString(), i.ToString()));
                                }
                            }
                        }
```

Slika 4.9: Koda za dinamično dodajanje podatkov v spustna menija za izbiro sobe in stene

```
<asp:RegularExpressionValidator ID="regexValidatorAudio" runat="server" ForeColor="Red"
style="left:490px; top:310px; position:absolute"
    ControlToValidate="FileUploadAudio"
    ErrorMessage="Le MP3 datoteke omogočene."
    ValidationExpression="(.*\.([Mm][Pp][3])|.*\.([Mm][Pp][3])$)">
</asp:RegularExpressionValidator>
```

Slika 4.10: Koda za preverjanje končnice audio datoteke

4.1.7 Urejanje eksponatov

Okno daje uporabniku možnost pregleda vnesenih eksponatov in spreminjanja njihovih atributov. Prikazani so v tabeli, kjer posamezna stran vsebuje podatke o desetih eksponatih. V primeru prikaza večjega števila se tabela samodejno razdeli v večje število strani. Med stranmi se lahko premikamo v spodnji navigacijski vrstici tabele.

V tabeli so prikazani vsi atributi, ki jih ima posamezen eksponat v podatkovni bazi. Določene lahko v tabeli sortiramo s klikom v glavi (angl. header) kolone. Uporabniku smo omogočili dodatno sortiranje eksponatov po razstavi. Nad tabelo je spustni meni z imeni razstav, ki so trenutno vnesene v podatkovno bazo. Glede na izbiro uporabnika se tabela dinamično spremeni. Če se uporabnik odloči za sortiranje eksponatov po določeni razstavi, se mu poleg spustnega menija prikaže obvestilo, katero razstavo je izbral. V primeru, da uporabnik izbere razstavo, ki nima vnesenih eksponatov, se prikaže le glava tabele.

Uporabnik ima poleg samostojnega vnosa znakov tudi možnost izbire preko spustnih menijev. V tabeli so trije spustni meniji, namenjeni spreminjanju slike in zvočne vsebine eksponata. Podatki se v spustne menije dodajajo dinamično ob vsakem prikazu strani. Pomagali smo si s knjižnico HtmlAgilityPack, ki se uporablja za lažji razcep (angl. split) podatkov iz dokumenta HTML. Slika 4.11 prikazuje uporabo kode, ki pridobi imena datotek shranjenih na spletnem strežniku. Razdelimo jih glede na njihove končnice. Datoteke, ki imajo končnico mp3, dodamo v seznam zvok, datoteke s končnicami jpg in png pa v seznam slike. Seznam zvok dodamo kot vir (angl. source) podatkov za spustni meni za spremembo zvočne vsebine eksponata, seznam slike pa se uporabi kot vir podatkov za spustni meni za spremembo slike eksponata.

Brisanje eksponata

Klik na gumb Delete omogoča uporabniku brisanje eksponata, ki je v tabeli levo od gumba. Ob njegovi uporabi se prikaže opozorilo (slika 4.12), ki

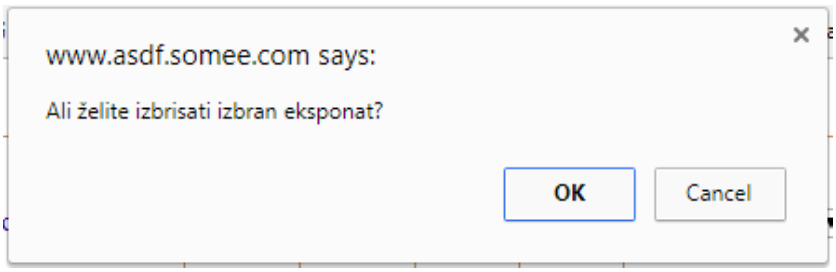
```
string baseURL = "http://asdf.somee.com/ImageStorage/";
WebClient client = new WebClient();
string content = client.DownloadString(baseURL);

var doc = new HtmlAgilityPack.HtmlDocument();
doc.LoadHtml(content);

slike.Add("");
audio.Add("");
foreach (HtmlNode node in doc.DocumentNode.SelectNodes("//a"))
{
    string temp = node.ChildNodes[0].InnerHtml;
    if (temp.Contains("jpg") || temp.Contains("jpeg") || temp.Contains("png"))
    {
        slike.Add(node.ChildNodes[0].InnerHtml);
    }
    else if (temp.Contains("mp3"))
    {
        audio.Add(node.ChildNodes[0].InnerHtml);
    }
}
```

Slika 4.11: Koda, ki pridobi imena datotek shranjenih na spletnem strežniku

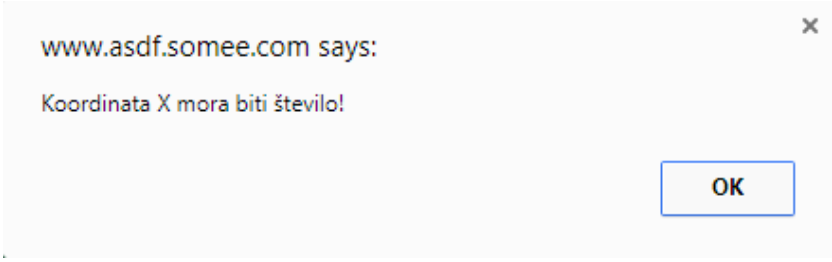
uporabnika vpraša, ali želi izbran eksponat res izbrisati. Če uporabnik izbere gumb OK, se eksponat izbriše. Ob kliku na gumb Cancel se operacija brisanja prekine.



Slika 4.12: Opozorilo o brisanju eksponata

Posodabljanje eksponata

Klik na gumb Edit omogoča urejanje atributov posameznega eksponata. Ko spremenimo atribut ali več atributov kliknemo gumb Update, ki nam posodobi podatke o eksponatu. Če vneseni podatki ustrezajo zahtevam, se podatki o eksponatu posodobijo. V primeru napačno vpisanih podatkov se uporabniku prikaže obvestilo o napaki in podatki o eksponatu se ne posodobijo. Slika 4.13 prikazuje primer obvestila o napaki pri spreminjanju koordinate X. Uporabnik je uporabil črke, medtem ko so dovoljeni znaki le števila.



Slika 4.13: Obvestilo o napačno vpisanih podatkih pri spremembi koordinate X

4.1.8 Prikaz trenutnih razstav

Uporabnik ima možnost pregleda in spreminjanja podatkov že vnesenih razstav ter možnost brisanja posamezne razstave. Podatki o razstavah so ravno tako prikazani v tabeli, ki je bila opisana v poglavju 4.1.7. Ker gre za isto vrsto tabele ima tudi tukaj uporabnik možnost sortiranja po določenem atributu. Na sliki 4.14 je vidna tabela, ki prikazuje vnesene razstave.

Na levi strani tabele (slika 4.14a) so prikazana imena, avtorji in vstopna gesla razstav. Vse razen zadnje razstave imajo ista vstopna gesla. Gre namreč za stalno zbirko razstav v Narodni galeriji. Atribut *slika*, ki ga vidimo na sliki 4.14b pove, kateri logotip trenutno predstavlja razstavo. Atribut

ime	avtor	geslo
1200-1600	Janez Akvila, Osbalt Kitell, Catarino Veneziano, Giovanni Francesco da Rimini, Marx Reichlich,	dva
1600-1700	Giuseppe Vicenzino, Peter van Kessel, Pietro Ricchi, Gregorio Lazzarini, Felice Ficherelli, Bartolomeo Bettera, Almanach	dva
1700-1800	Martino Altomonte, Domenico Brandi, Anton Cebej, Nicola Grassi, Franc Jelovšek, Jožef Straub	dva
1800-1820	Moritz Michael Daffinger, Anton Dietrich, Franc Kavčič/Caucig	dva
1820-1870	Marija Auersperg Attems, Anton Karinger, Matevž Langus, Marko Pernhart, Jožef Tominc	dva
1870-1900	Anton Ažbe, Vlaho Bukovac, Alojz Gangl, Anton Gvajc, Jožef Petkovšek, Janez Šubic	dva
1900-1918	Franc Berneker, Lojze Dolinar, Ivan Grohar, Rihard Jakopič, Matija Jama, Ivan Vavpotič	dva
Od 1918 dalje	Stojan Batič, Massimo Campigli, Tone Kralj, Peter Loboda, France Mihelič, Jože Plečnik	dva
Zoran Mušič	Zoran Mušič	dva
Pot sejalca	brez	a a
	1 2	

(a) Leva stran tabele

slika	Spremeni sliko	aktivna	
www.asdf.somee.com/ImageStorage/srednjiVek.jpg		☑	Edit Delete
www.asdf.somee.com/ImageStorage/maneirizem.jpg		☑	Edit Delete
www.asdf.somee.com/ImageStorage/barok.jpg		☑	Edit Delete
www.asdf.somee.com/ImageStorage/neoklasicizem.jpg		☑	Edit Delete
www.asdf.somee.com/ImageStorage/bidermajer.jpg		☑	Edit Delete
www.asdf.somee.com/ImageStorage/romantika.jpg		☑	Edit Delete
www.asdf.somee.com/ImageStorage/intimizem.JPG		☑	Edit Delete
www.asdf.somee.com/ImageStorage/tretja.jpg		☑	Edit Delete
www.asdf.somee.com/ImageStorage/zoran.jpg		☑	Edit Delete
www.asdf.somee.com/ImageStorage/juhu.jpg		☑	Edit Delete

(b) Desna stran tabele

Slika 4.14: Tabela o trenutno vnešenih razstavah

Spremeni sliko daje uporabniku možnost, da zamenja logotip razstave, ki je izbran v levi koloni. Atribut *aktivna* predstavlja stanje razstave. Gre za bitno vrednost. Ob vnosu nove razstave se samodejno nastavi na 1. Vrednost 1 pomeni, da je razstavo aktivna. V primeru, da razstava ni aktivna in obiskovalec pozna vstopno geslo, se to šteje kot napačno. Uporabnik ima ravno tako možnost spreminjanja atributov ali brisanja razstave.

4.2 Komunikacija Spletni strežnik – Android

Za uspešno komunikacijo mobilne aplikacije s strežnikom je bilo potrebno spletno aplikacijo definirati kot storitev RESTful.

Sprememba konfiguracijske datoteke

Za delovanje spletne aplikacije kot storitev RESTful, moramo spremeniti del kode o konfiguraciji projekta, ki je v datoteki Web.config (slika 4.15). Spremeniti moramo vrednost značke endpoint address. BehaviorConfiguration spremenimo v "restfulBehavior". Značko binding spremenimo v "webHttpBinding".

```
<system.serviceModel>
  <services>
    <service name="SpletnaAplikacija.RestServiceImpl">
      <endpoint address="" behaviorConfiguration="restfulBehavior" binding="webHttpBinding"
      bindingConfiguration="" name="Service1" contract="SpletnaAplikacija.IRestServiceImpl"/>
      <host>
        <baseAddresses>
          <add baseAddress="http://localhost/service1"/>
        </baseAddresses>
      </host>
    </service>
  </services>
  <behaviors>
    <endpointBehaviors>
      <behavior name="restfulBehavior">
        <webHttp/>
      </behavior>
    </endpointBehaviors>
```

Slika 4.15: Del kode konfiguracijske datoteke spletne aplikacije

Datoteka IRestServiceImpl.cs

V okviru projekta smo definirali dve datoteki: IRestServiceImpl.cs in RestServiceImpl.svc. Datoteka IRestServiceImpl.cs vsebuje definicije končnih točk in razrede z atributi, ki jih storitev REST vrača odjemalcu. V datoteki moramo definirati vse končne točke, ki jih bo naša aplikacija vsebovala.

```
[OperationContract]
[WebInvoke(Method = "GET",
ResponseFormat = WebMessageFormat.Json,
UriTemplate = "DobiIDNadstropja/{id}")]
1 reference | 0 changes | 0 authors, 0 changes
int dobiIDNadstropja(string id);
```

(a) Definicija metode DobiIDNadstropja

```
public class Nadstropja {
    [DataMember]
    1 reference | ufatur, 41 days ago | 1 author, 1 change
    public string ime { get; set; }
    [DataMember]
    1 reference | ufatur, 41 days ago | 1 author, 1 change
    public string tloris { get; set; }
}
```

(b) Definicija razreda Nadstropja, ki ga vrača metoda dobiIDNadstropja

Slika 4.16: Definicija končne točke in razreda z atributi, ki ga končna točka vrača odjemalcu

Slika 4.16a prikazuje definicijo metode DobiIDNadstropja. Slika 4.16b definira atribute razreda Nadstropja. Za vsak atribut moramo določiti tip (int, string, bool) in metodo, ki nastavi vrednost atributa, ter metodo, ki dobi vrednost atributa. V spodnjih alinejah bomo definirali, kaj pomenijo posamezne vrstice v definiciji metode.

- **[OperationContract]** – definiramo pred vsako metodo, ki bo dostopna odjemalcem.
- **[WebInvoke(Method = "GET")]** – določimo, da gre za metodo GET.
- **ResponseFormat = WebMessageFormat.Json** – določimo JSON kot format odgovora. Kot alternativo lahko izberemo tudi XML.
- **UriTemplate = "DobiIDNadstropja/{id}"** – določimo naslov URL, na katerem bo metoda dostopna odjemalcem.
- **int dobiIDNadstropja(string id)** – definicija razreda, ki izvede kodo metode DobiIDNadstropja.

Datoteka RestServiceImpl.svc

Datoteka RestServiceImpl.svc vsebuje kodo, ki se izvaja ob klicih storitev REST (slika 4.17). Koda se izvede ob klicu metode DobiIDNadstropja.

```
public int dobiIDNadstropja(string id)
{
    using(SqlConnection connection = new SqlConnection(connectionStrings)) {
        using(SqlCommand getData = new SqlCommand("SELECT DISTINCT(s.nadstropje_ID) FROM Eksponat e "+
        "INNER JOIN Razstava r on e.razstava_ID = r.id_razstava INNER JOIN Soba s on e.soba_ID =" +
        "s.id_soba WHERE r.id_razstava = '" + id + "'", connection)) {
            connection.Open();
            object data = getData.ExecuteScalar();
            connection.Close();
            if(data != null) {
                return Int32.Parse(data.ToString());
            }
            return -1;
        }
    }
}
```

Slika 4.17: Koda, ki se izvede ob klicu metode DobiIDNadstropja

Seznam končnih točk

Za delovanje mobilne aplikacije smo morali definirati končne točke, ki so predstavljene v nadaljevanju.

LoginID/{password} vrne število razstav, ki imajo vstopno geslo enako parametru *password* in atribut *aktivna* postavljen na 1. V nasprotnem primeru se vrne vrednost –1.
Razstave/{geslo} vrne podatke o razstavah z geslom, ki je podan kot vhodni parameter.
DobiSlikoNadstropje/{id} vrne ime in tloris nadstropja glede na podan ID nadstropja.
DobiSlikoSobe/{id} vrne podatke o sobi s podanim ID-jem, ter nadstropje, v katerem se soba nahaja.
StNadstropij vrne število nadstropij.
DobiEksponate/{gesloRazstava}/{idSoba}/{idStena} vrne podatke

o eksponatih, ki spadajo v razstavo z vstopnim geslom podanim kot parameter *geslo Razstava* in so v sobi z ID-jem *idSoba* in steni z ID-jem *idStena*.
DobiKipe/{gesloRazstava}/{idSoba} vrne podatke o kipih, ki spadajo v razstavo z vstopnim geslom podanim kot parameter gesloRazstava in so v sobi z ID-jem idSoba.
DobiInfoEksponat/{id} vrne vse podatke o eksponatu z ID-jem *id*.
DobiIDNadstropja/{id} vrne ID nadstropja, v katerem je razstava *id*. V primeru, da razstava nima vnesenih eksponatov, se vrne vrednost –1.

4.3 Ogled razstave

Mobilna aplikacija je izdelana za pametne telefone z operacijskim sistemom Android. Namenjena je obiskovalcem Narodne galerije. Uporaba mobilne aplikacije da obiskovalcu večjo svobodo pri obisku Narodne galerije, saj ta dobi informacije o razstavljenem eksponatu v katerem koli delu sobe. Zaradi tega obiskovalcem ni treba čakati pred vsakim eksponatom, da lahko preberejo opis na steni.

4.3.1 Vstopna stran

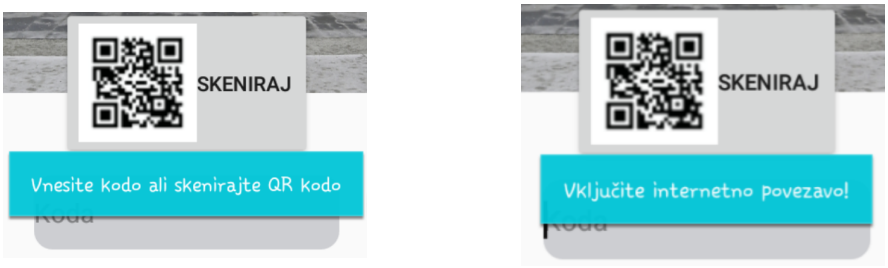
Vstopna stran mobilne aplikacije prikazuje osnovne informacije in povezave. Uporabnik se lahko prijavi z vpisom vstopnega gesla v tekstovno polje ali skeniranjem QR-kode. V primeru, da se uporabnik želi v mobilno aplikacijo prijaviti brez vstopnega gesla, se mu prikaže obvestilo (slika 4.18a).

4.3.2 Internetna povezava

Za delovanje mora imeti naprava ves čas omogočeno povezavo s spletnim strežnikom, dokler je aplikacija v uporabi. V projektu smo definirali metodo haveNetworkConnection[1] (slika 4.19), ki preverja, ali ima naprava vzposta-

[1]https://stackoverflow.com/questions/9002180/using-context-connectivity-service-to-check-for-internet-connection

vljeno povezavo z internetom. Metodo kličemo pred vsakim dostopom do storitev REST za pridobitev podatkov o aktivnih povezavah v napravi. V primeru, da katera povezava vsebuje ime "WIFI" ali "MOBILE" metoda vrne logično vrednost true, kar pomeni, da je naprava povezana na internet in lahko začne komunikacijo s storitvijo REST. V nasprotnem primeru metoda vrne logično vrednost false. Komunikacija s storitvijo REST je prekinjena. Uporabniku se prikaže obvestilo, naj napravo poveže na internet (slika 4.18b).



(a) Obvestilo o možnostih prijave v mobilno aplikacijo

(b) Obvestilo o vzpostavitvi povezave z internetom

Slika 4.18: Obvestila o napakah

4.3.3 Izbira jezika

Mobilna aplikacija ima vgrajeno možnost prikaza podatkov v različnih jezikih. V trenutni izvedbi sta podprta slovenščina (slika 4.20a) in angleščina (slika 4.20b). Jezik, ki ga bo mobilna aplikacija uporabila, je odvisen od jezika naprave uporabnika. V primeru, da ima uporabnik kot jezik svojega mobilnega telefona nastavljeno slovenščino, bo aplikacija delovala v slovenskem jeziku. V primeru, da je jezik naprave angleški, oziroma kateri drugi jezik, bo aplikacija delovala v angleškem jeziku.

Za dodajanje novega jezika v programu Android Studio moramo dodati novo datoteko strings.xml. Pri ustvarjanju datoteke moramo izbrati jezik, pri katerem bo datoteka aktivna. Datoteka strings.xml vsebuje značke z imeni posameznih tekstov. Vrednost posamezne značke moramo zapisati v jeziku,

```
private boolean haveNetworkConnection() {
    boolean haveConnectedWifi = false;
    boolean haveConnectedMobile = false;

    ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo netInfo = cm.getActiveNetworkInfo();
    if(netInfo != null){
        if (netInfo.getTypeName().equalsIgnoreCase("WIFI")) {
            if (netInfo.isConnected()) {
                haveConnectedWifi = true;
            }
        }

        if (netInfo.getTypeName().equalsIgnoreCase("MOBILE")) {
            if (netInfo.isConnected()) {
                haveConnectedMobile = true;
            }
        }
    }
    return haveConnectedWifi || haveConnectedMobile;
}
```

Slika 4.19: Metoda haveNetworkConnection, ki preverja ali je naprava povezana na internet

ki ga želimo dodati. Del kode datoteke strings.xml prikazuje slika 4.21a. V projektu imamo lahko toliko datotek strings.xml, kolikor želimo imeti podprtih jezikov. V primeru, da dodamo jezik, moramo nadgraditi spletno aplikacijo, da se lahko vnašajo podatki o eksponatih v novem jeziku. V vsaki datoteki strings.xml moramo imeti ista imena značk. V nasprotnem primeru aplikacija ne bo delovala.

4.3.4 Skeniranje QR-kode

Uporabnik se lahko poleg vpisa vstopnega gesla v aplikacijo prijavi tudi s skeniranjem QR-kode s klikom na gumb Skeniraj. Za skeniranje se uporablja knjižnica ZXing in mobilna aplikacija Barcode Scanner.V takem primeru aplikacija preveri, ali ima uporabnik že nameščeno aplikacijo Barcode Scanner. V primeru, da je aplikacija nameščena, lahko uporabnik skenira QR-kodo. V nasprotnem primeru aplikacija uporabniku ponudi prenos aplikacije s strani Google Play.

Ko uporabnik skenira kodo, se izvede metoda onActivityResult (slika 4.22).

(a) Prva stran (slovenski jezik)

(b) Prva stran (angleški jezik)

Slika 4.20: Prva stran mobilne aplikacije glede na izbran jezik uporabnika na mobilnem telefonu

Če je koda rezultata skeniranja enaka "RESULT˙OK"pomeni, da je bilo skeniranje uspešno. V spremenljivko *contents* dobimo podatke, ki jih je vsebovala QR-koda (vstopno geslo razstave). V primeru, da je geslo sestavljeno iz več besed, mu spremenimo obliko, da ga storitev REST pravilno interpretira. Pravilnost vstopnega gesla preverimo s klicem na storitev REST LoginID.

4.3.5 Metodi doInBackground in onPostExecute

Za dostop do storitev REST uporabljamo zasebne razrede, ki razširjajo razred AsyncTask. Razred se mora povezati s spletno storitvijo, prenesti podatke in jih prikazati. Izvaja se asinhrono in s tem omogoča uporabniku, da ves čas izvajanja nadaljuje s svojim delom. Posamezni razred, ki razširja razred AsyncTask je sestavljen iz metod doInBackground in onPostExecute. Naloga metode doInBackground je povezava do storitve REST in prenos podatkov. Na sliki 4.23 je prikazana metoda doInBackground, ki se izvede ob prijavi v

```
<string name="playstore">Namestite PlayStore</string>
<string name="vpisanakoda">Vpisana koda je napačna!</string>
<string name="internet">Vključite internetno povezavo!</string>
<string name="iskanjeSobe">Išči sobo</string>
<string name="razstave">Razstave</string>
<string name="napacnaSlika">Slika s podanim ID-jem ne obstaja!</string>
<string name="napacnaSoba">Soba s podanim ID-jem ne obstaja!</string>
<string name="sobaNaslov">Soba </string>
```

(a) Del kode datoteke strings.xml za slovenski jezik

```
<string name="playstore">Install PlayStore First</string>
<string name="vpisanakoda">Code you wrote is wrong!</string>
<string name="internet">Please connect to the Internet!</string>
<string name="iskanjeSobe">Search room</string>
<string name="razstave">Exhibitions</string>
<string name="napacnaSlika">Picture with this ID does not exist!</string>
<string name="napacnaSoba">Room with this ID does not exist!</string>
<string name="sobaNaslov">Room </string>
```

(b) Del kode datoteke strings.xml za angleški jezik

Slika 4.21: Datoteka strings.xml

```
public void onActivityResult(int requestCode, int resultCode, Intent intent) {
    if(haveNetworkConnection()){
        if (requestCode == 0) {
            if (resultCode == RESULT_OK) {
                String contents = intent.getStringExtra("SCAN_RESULT");
                temp = contents;
                if(contents.matches(".*\\s+.*")){
                    contents = contents.replaceAll("\\s+", "%20");
                }
                String url = "http://asdf.somee.com/RestServiceImpl.svc/LoginID/"  + contents;
                RESTCallTask task = new RESTCallTask();
                task.execute(url);
                // Handle successful scan
            } else if (resultCode == RESULT_CANCELED) {
                // Handle cancel
            }
        }
    } else {
        if (myToast != null){
            myToast.cancel();
        }
        myToast = Toast.makeText(getApplicationContext(), "Please connect to the Internet!", Toast.LENGTH_SHORT);
        myToast.show();
    }
}
```

Slika 4.22: Metoda onActivityResult, ki se kliče ob zaključku skeniranja

mobilno aplikacijo.

```
@Override
protected String doInBackground(String... params) {
    URL url;
    HttpURLConnection urlConnection = null;
    try {
        url = new URL(params[0]);
        urlConnection = (HttpURLConnection) url.openConnection();

        int responseCode = urlConnection.getResponseCode();
        if (responseCode == HTTP_OK) {
            InputStream in = urlConnection.getInputStream();
            String result = readStream(in);
            System.out.println("From server: " + result);
            return result;
        } else {
            System.out.println("Response code: " + responseCode);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (urlConnection != null) {
            urlConnection.disconnect();
        }
    }
    return null;
}
```

Slika 4.23: Metoda doInBackground

Metoda kot parameter dobi URL-naslov storitve REST, na katero se mora povezati. Ko pridobi podatke, preverimo kodo odgovora. Če je koda enaka "HTTP_OK" pomeni, da se je storitev REST izvedla in je odjemalec pridobil podatke. Podatke preberemo s pomočjo metode ReadStream.

Metoda onPostExecute se uporablja za delo s podatki, ki jih je pridobila metoda doInBackground. Slika 4.24 prikazuje metodo onPostExecute, ki se uporablja pri prijavi v mobilno aplikacijo. V spremenljivko *result* shranimo odgovor storitve REST. V spremenljivko *vpisana* shranimo vstopno geslo, ki ga je uporabnik vnesel v vnosno polje koda. Če vrednost spremenljivke ni enaka praznemu nizu pomeni, da je bilo vstopno geslo pravilno. Ker bomo

vstopno geslo v nadaljevanju še potrebovali, ga shranimo v razred Bundle. Razred Bundle se uporablja za prenos podatkov med različnimi aktivnostmi aplikacij Android.

```
@Override
protected void onPostExecute(String result){
    result = result.replaceAll("[^-?0-9]+", " ").trim();
    String vpisana = String.valueOf(koda.getText());
    if(vpisana.equals("")){
        vpisana = temp;
    }
    if(vpisana.matches(".*\\s+.*")){
        vpisana = vpisana.replaceAll("\\s+", "%20");
    }
    if(result.equals("-1")){
        if (myToast != null){
            myToast.cancel();
        }
        myToast = Toast.makeText(getApplicationContext(),
                getResources().getString(R.string.vpisanakoda), Toast.LENGTH_LONG);
        myToast.show();
    } else {
        koda.setText("");
        Intent intent = new Intent(getApplicationContext(), Razstave.class);
        Bundle b = new Bundle();
        b.putString("koda", vpisana);
        intent.putExtras(b);
        startActivity(intent);
    }
}
```

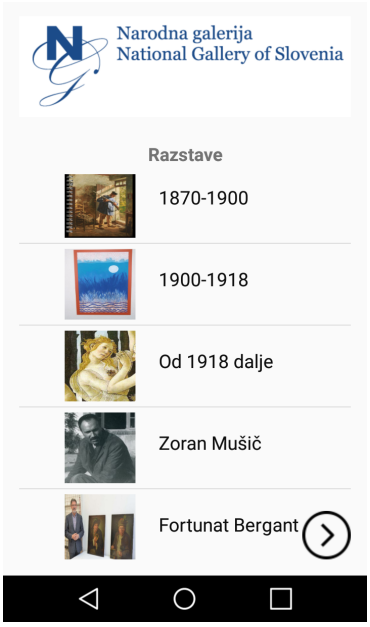
Slika 4.24: Metoda onPostExecute

4.3.6 Aktivnost 'razstave'

Aktivnost 'razstave' nam prikaže seznam razstav, ki si jih lahko ogledamo z vpisanim vstopnim geslom. Vsaka razstava je prikazana s svojim logotipom in imenom razstave. Primer seznama razstav za vstopno geslo "SZumetnin"je prikazan na sliki 4.25b. S klikom na spodnji desni gumb se uporabniku prikaže aktivnost 'nadstropja'. Ob kliku na določeno razstavo se izvede preusmeritev na aktivnost 'nadstropja'. Razlika je le, da mu ob kliku na določeno razstavo prikažemo nadstropje, kjer je ta razstavljena, v nasprotnem primeru

pa tloris pritličja.

(a) QR-koda z vstopnim geslom “SZumetnin”



(b) Seznam razstav za vstopno geslo “SZumetnin”

Slika 4.25: Podatki za ogled razstave

Seznam razstav dobimo s klicem na eno izmed storitev REST. Odgovor, ki ga dobimo je v formatu JSON. Za njihovo uporabo jim moramo spremeniti format zapisa. V metodi onPostExecute spremenimo format z uporabo tabele JSON (angl. JSONArray) in objektov JSON. Do posameznega atributa lahko dostopamo z metodo getString, ki nam glede na ključ vrne njegovo vrednost. Dobljene atribute shranimo v seznam *data*. Del kode metode onPostExecute je prikazan na sliki 4.26.

4.3.7 Aktivnost ’nadstropje’

Aktivnost ’nadstropje’, prikazana na sliki 4.27a, je osrednja aktivnost mobilne aplikacije. Uporabniku daje možnost pregleda sob v posameznih nadstropjih. Večji del okna zajema tloris posameznega nadstropja. Uporabnik se med nadstropji lahko premika z gumboma, ki sta na desni strani zaslona.

```
for(int i = 0; i < jsonArray.length(); i++){
    try {
        JSONObject object = jsonArray.getJSONObject(i);
        String idR = object.getString("id_razstava");
        String ime = object.getString("ime");
        String slika = object.getString("slika");
        slika = slika.replace("\\", "");
        slika = "http://" + slika;
        data.add(new Razstava(ime, slika, idR));
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Slika 4.26: Del kode metode onPostExecute, ki spremeni format zapisa podatkov

Nad tlorisom je izpisano ime nadstropja. S tem uporabnik natančno ve, katero nadstropje Narodne galerije ima prikazano. V primeru, da je uporabniku prikazano najnižje oz. najvišje nadstropje se gumbu dol oz. gor spremeni barva in ga ni mogoče klikniti. V tlorisu nadstropja so sobe, v katerih so razstavljeni eksponati označene s števili. Posamezno število predstavlja ID sobe v podatkovni bazi. Uporabnik lahko v sobo dostopa s skeniranjem QR-kode, ki je postavljena ob vhodu v sobo ali vpisom ID-ja sobe v tekstovno polje, ki je v zgornjem levem delu zaslona. Ob kliku na gumb Nazaj se uporabniku prikaže obvestilo o izhodu z aplikacije (slika 4.27b). Ob kliku na gumb DA je uporabnik izpisan iz aplikacije in se mu prikaže prijavna stran. V primeru, da klikne gumb NE ostane prijavljen še naprej.

Knjižnica Picasso

Za prikaz slik v mobilni aplikaciji smo uporabili knjižnico Picasso[2]. Knjižnica je popularna predvsem zaradi preprostosti (večino ukazov napišemo v eni vrstici). Samodejno poskrbi za večino pasti pri prikazu slik. Prikaz slik

[2]http://square.github.io/picasso/

(a) Prikaz aktivnosti nadstropje (b) Obvestilo ob izhodu

Slika 4.27: Prikaz aktivnosti

nadstropij se izvede v metodi (slika 4.28) z enostavno sintakso:

- Picasso.with(this) – definiramo aktivnost, ki bo prikazala sliko.
- .load("http://"+ tloris) – URL slike, ki jo želimo prikazati. Če pred naslov URL ne vpišemo "http://"knjižnica vrne napako.
- .fit() – izberemo, če želimo, da se slika prilega prostoru, kjer bo prikazana
- .into(imageView) – definiramo objekt za katerega želimo, da nam prikaže sliko

4.3.8 Aktivnost 'soba'

Slika 4.29 prikazuje aktivnost 'soba', do katere lahko uporabnik dostopa s skeniranjem QR-kode ob vhodu v prostor ali z vpisom ID-ja sobe v tekstovno

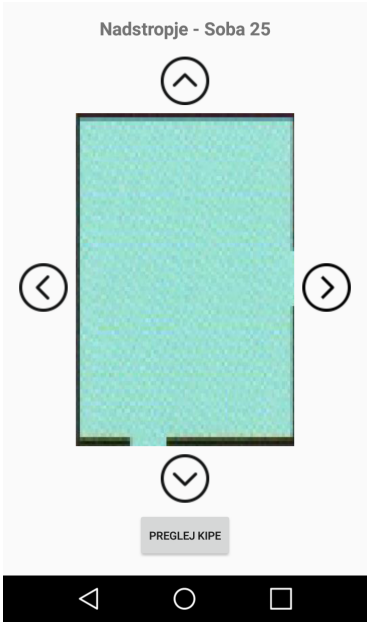
```
private void slika(String tloris, String ime){
    Picasso.with(this).load("http://" + tloris).fit().into(imageView);
    pAttacher = new PhotoViewAttacher(imageView);
    pAttacher.update();
    naslovNadstropja.setText(ime);
}
```

Slika 4.28: Metoda slika, ki poskrbi za prikaz slike

polje. Večji del zaslona pokriva slika tlorisa sobe. Dogovorili smo se, da bomo tloris obrnili tako, da bo eden izmed vhodov na spodnjem delu slike. Glede na to, da imajo nekatere sobe več vhodov, bi lahko tloris shranili glede na vse možne vhode. Dogovorili smo se, da bodo vse sobe imele največ štiri stene. Dejansko imajo nekatere sobe v galeriji lahko tudi več sten. Pri takšnih sobah smo število sten poenostavili. Več manjših sten, ki so obrnjene v isto smer, smo definirali kot eno steno. S klikom na določen gumb ob izbrani steni dobimo informacije o eksponatih, ki so tam razstavljeni. Na vrhu zaslona dobi uporabnik podatke o ID–ju sobe, ter nadstropja v katerem se soba nahaja. Na dnu zaslona je gumb "Preglej kipe". S klikom nanj dobi uporabnik informacije o razstavljenih kipih v sobi.
Razlika med gumbom "Preglej kipe"in gumbom za pregled eksponatov na določeni steni je v klicu storitve REST. Storitvi za stene in kipe sta podobni, razlika je le v spremenjeni poizvedbi v podatkovni bazi.

4.3.9 Aktivnost 'eksponat'

Aktivnost prikaže eksponate, razstavljene na določeni steni ali po prostoru. Posamezen eksponat je predstavljen s sliko, imenom eksponata in imenom avtorja. V seznamu so privzeto razporejeni po položaju na steni. Sortiramo jih glede na koordinato X, ki je bila določena ob vnosu posameznega eksponata v podatkovno bazo. Slika 4.30a prikazuje seznam eksponatov za sobo z ID-jem 25 in steno 1. Na dnu zaslona je gumb "Sortiraj od a–z". Ob kliku na gumb eksponate sortiramo po abecednem vrstnem redu (slika 4.30b). Ob vsakem kliku na gumb se spremeni besedilo na njem. Do informacij o posameznem



Slika 4.29: Primer aktivnosti soba (ID = 25)

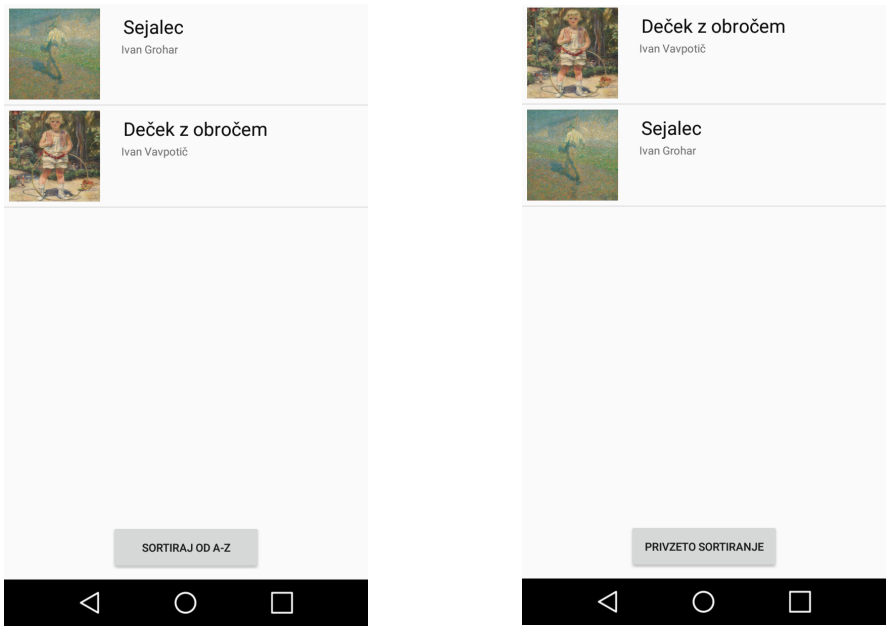
eksponatu pridemo s klikom na polje seznama.

Razred Data

Slika 4.31 prikazuje razred Data, ki se uporablja za shranjevanje večjega števila eksponatov v obliki objektov. Posamezen eksponat ima atribute *id*, *ime*, *avtor*, *link* in atribut *x*, ki se nanaša na koordinato X na steni. Pri razredu Data smo definirali, da implementira razred Parceleable. S tem poskrbimo, da lahko v metodi onSaveInstanceState, ki se uporablja za prenos podatkov med rotiranjem zaslona, prenašamo objekte tipa seznam (angl. ArrayList). S tem omogočimo uporabniku, da se ob morebitni rotaciji zaslona ne bo spremenil pogled seznama sortiranih eksponatov.

Razred CustomListAdapter

Razred CustomListAdapter (slika 4.32) se uporablja za prikaz eksponatov. Vsebuje atributa *context* in *itemname*. Atribut *context* definira trenutni



(a) Seznam eksponatov na steni 1 (b) Sortiranje eksponatov

Slika 4.30: Eksponati v sobi ID = 25

```
class Data implements Parcelable {
    String id;
    String ime;
    String avtor;
    String link;
    int x;

    public Data(String id, String prvi, String s, String drugi, int pos){
        this.id = id;
        this.ime = prvi;
        this.avtor = s;
        this.link = drugi;
        this.x = pos;
    }

    protected Data(Parcel in) {
        id = in.readString();
        ime = in.readString();
        avtor = in.readString();
        link = in.readString();
        x = in.readInt();
    }
```

Slika 4.31: Razred Data

pogled, atribut *itemname* pa seznam eksponatov, ki jih želimo prikazati. Metoda getView poskrbi za napolnitev pogleda seznama (angl. ListView). Preden prikažemo eksponate, moramo definirati videz polja pogleda seznama, ki bo prikazovalo posamezen eksponat. Polje smo definirali v datoteki mylist.xml.

```
public class CustomListAdapter extends ArrayAdapter<Data> {

    private final Activity context;
    private final ArrayList<Data> itemname;

    public CustomListAdapter(Activity context, ArrayList<Data> itemname) {
        super(context, R.layout.mylist, itemname);

        this.context=context;
        this.itemname=itemname;
    }

    public View getView(int position, View view, ViewGroup parent) {
        LayoutInflater inflater = context.getLayoutInflater();
        View v = inflater.inflate(R.layout.mylist, parent, false);

        TextView txtTitle = (TextView) v.findViewById(R.id.item);
        TextView avtor = (TextView) v.findViewById(R.id.textView1);
        ImageView imageView = (ImageView) v.findViewById(R.id.icon);

        txtTitle.setText(itemname.get(position).ime);
        avtor.setText(itemname.get(position).avtor);
        Picasso.with(context).load(itemname.get(position).link).fit().into(imageView);

        return v;
    }
}
```

Slika 4.32: Razred CustomListAdapter

Gumb za sortiranje

Gumb za sortiranje je zelo praktičen v primeru velikega števila eksponatov na določeni steni. Privzeto sortiranje, ki se izvede tudi ob prikazu eksponatov, je sortiranje po X-koordinati eksponatov. Koordinata X predstavlja položaj eksponata na steni, ali pa v prostoru, če je eksponat kip. Dogovorili smo se, da bodo eksponati z manjšo koordinato X na steni postavljeni bolj levo, posledično pa eksponati z večjo koordinato X bolj desno. Uporabnik ima možnost sortiranja tudi po abecednem vrstnem redu. Koda, ki se izvede ob pritisku na gumb za sortiranje, je prikazana na sliki 4.33. Pri sortiranju

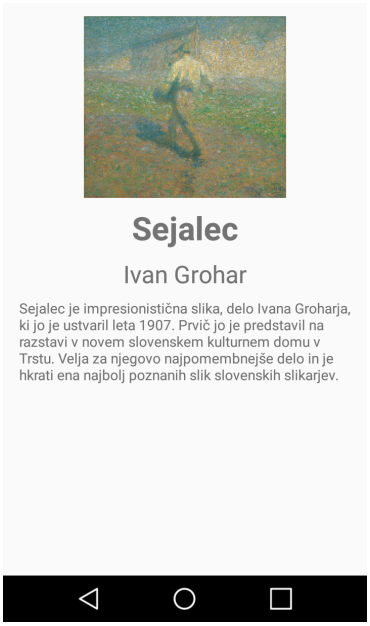
je pomembna metoda parseEksponate. Kličemo jo s tremi atributi. Atribut *seznamEksponatov* predstavlja seznam eksponatov, ki jih želimo prikazati. Atribut *funkcija* predstavlja število, s katerim povemo, kako želimo eksponate sortirati. Atribut *kdaj* je logična vrednost, ki pove, ali se funkcija kliče ob začetku dejavnosti ali ob pritisku na gumb Sortiraj. Ker moramo ob vsakem pritisku na gumb Sortiraj eksponate drugače sortirati, po koncu metode parseEksponate zamenjamo vrednost atributa *funkcija*.

```
abeceda.setOnClickListener((v) → {
    parseEksponate(seznamEksponatov, funkcija, kdaj);
    funkcija = 3 - funkcija;
    if(haveNetworkConnection()) {
        CustomListAdapter adapter = new CustomListAdapter(SeznamEksponatov.this, data);
        list.setAdapter(adapter);
    } else {
        if (myToast != null){
            myToast.cancel();
        }
        myToast = Toast.makeText(getApplicationContext(),
                getResources().getString(R.string.internet), Toast.LENGTH_SHORT);
        myToast.show();
    }
});
```

Slika 4.33: Akcija, ki se izvede ob kliku na gumb za sortiranje eksponatov

4.3.10 Prikaz posameznega eksponata

Aktivnost *Prikaz* posameznega eksponata prikaže obiskovalcu informacije o posameznem eksponatu. Za primer slike Sejalec (slika 4.34a) so prikazane slika, ime in avtor eksponata. Večji del zaslona prekriva tekstovni del z osnovnimi informacijami v slovenskem jeziku. V spodnjem delu zaslona je gumb za poslušanje zvočne vsebine. Gumb prikažemo glede na to, ali ima eksponat v podatkovni bazi vključeno zvočno datoteko ali ne (slika 4.34b).

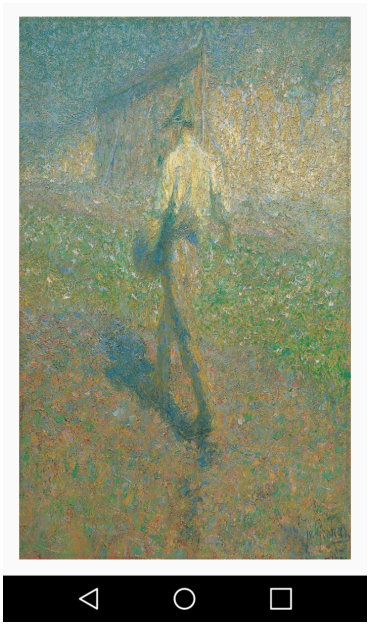


(a) Slika Sejalec

(b) Kopija freske Mrtvaški ples

Slika 4.34: Prikaz eksponata, desna slika vsebuje audio vsebino, leva pa ne

S klikom na sliko se obiskovalcu pokaže celozaslonska slika eksponata (slika 4.35a). Ob kliku na gumb Poslušaj se začne predvajati zvočna vsebina o eksponatu. Obiskovalec lahko kadarkoli konča predvajanje s pritiskom na gumb za zaustavitev, ki je poleg gumba Poslušaj (slika 4.35b).



(a) Celozaslonski pogled

(b) Prikaz gumba za zaustavitev poslušanja

Slika 4.35: Prikaz eksponata

Poglavje 5

Sklepne ugotovitve

Cilji, ki smo si jih zastavili pri izdelavi diplomske naloge, so bili doseženi. Funkcionalnosti, ki smo jih definirali pred začetkom načrtovanja, smo med implementacijo diplomske naloge še nekoliko nadgradili in uspešno realizirali. Aplikaciji delujeta in sta bili testirani z resničnimi podatki in slikami, ki se nahajajo na spletu, z dovoljenjem vodstva Narodne galerije. Spletna aplikacija omogoča prijavo v sistem, dodajanje novih vnosov podatkov o razstavah in eksponatih v podatkovno bazo in spreminjanje že obstoječih elementov v podatkovni bazi. Narejena je le v slovenskem jeziku. Mobilna aplikacija je namenjena obiskovalcem. Omogoča jim, da pridobijo informacije o posameznem eksponatu. Aplikacija omogoča skeniranje QR-kod, prikazovanje slik in teksta posameznih eksponatov in možnost poslušanja zvočnih vsebin. Podatki so shranjeni v dveh jezikih – v slovenščini in v angleščini. V mobilni aplikaciji se prikažejo v jeziku, ki je izbran na napravi. Načrti strani so nastali pred samo implementacijo diplomske naloge, zato so pri nekaterih slikah načrtov večja odstopanja v primerjavi s končno vizualno rešitvijo. Nekatera okna končne verzije spletne aplikacije sploh nimajo načrtov strani, saj so nastajala med implementacijo in jih na samem začetku nismo predvideli.

Trend uporabe mobilnih aplikacij je v vzponu. Večje svetovne galerije imajo svoje mobilne aplikacije, ki ponudijo obiskovalcem boljšo izkušnjo pri samem obisku. Končni izdelek diplomske naloge je prototip aplikacije, name-

njene ogledu galerij, zato bi bilo smiselno nekatere stvari izboljšati. Možni popravki:

- vizualni videz spletne aplikacije – izdelava nove prve strani, preglednejši in optimiziran videz tabel za pregled vnesenih razstav in eksponatov (prikazovanje manjšega števila atributov),
- registracija v spletni aplikaciji – potrebno je dodati dodatno preverjanje, ali je novi uporabnik res zaposlen v galeriji,
- dejansko število sten – posamezna soba naj bi imela dejansko število sten in ne največ štiri, kot smo predpostavili v naši izvedbi. Pod tlorisom sobe bi bilo toliko gumbov, kolikor sten ima soba (vsak gumb namenjen svoji steni),
- delovanje aplikacije v večjem številu jezikov – v aplikaciji bi lahko dodali še druge svetovne jezike (italijanščina, nemščina, francoščina),
- uporabniku bi lahko ponudili možnost premikanja po galeriji tudi s tehnologijami, kot so BLE (Bluetooth Low Energy) in NFC (Near Field Communication),
- za eksponate, postavljene po prostoru, bi bilo smiselno dodati več slik iz različnih zornih kotov,
- migracija strežnika za dejansko uporabo – manj omejitev glede velikosti naloženih datotek, možna večja velikost projekta.

Literatura

[1] Json - rfc dokument. Dosegljivo: http://www.ietf.org/rfc/rfc4627.txt. [Dostopano: 2. 8. 2017].

[2] Kaj je android. Dosegljivo: http://slo-android.si/prispevki/kaj-je-android.html. [Dostopano: 8. 8. 2017].

[3] Knjižnica zxing. Dosegljivo: https://github.com/zxing/zxing. [Dostopano: 13. 7. 2017].

[4] Narodna galerija. Dosegljivo: http://www.ng-slo.si/si/razstave-in-projekti/razstava/prenovljeni-narodni-dom?id=3723. [Dostopano: 12. 8. 2017].

[5] Ntc hosting - sql. Dosegljivo: https://www.ntchosting.com/encyclopedia/databases/structured-query-language/. [Dostopano: 2. 8. 2017].

[6] Faraz Rasheed. C# school, 2006. Dosegljivo: http://www.webcitation.org/6h8p6fWJk [Dostopano: 8. 8. 2017].

[7] Search micro services - rest. Dosegljivo: http://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer. [Dostopano: 1. 8. 2017].

[8] Search micro services - xml. Dosegljivo: http://searchmicroservices.techtarget.com/definition/XML-Extensible-Markup-Language. [Dostopano: 13. 8. 2017].

[9] Search mobile computing. Dosegljivo: http://searchmobilecomputing.techtarget.com/definition/Android-OS. [Dostopano: 31. 7. 2017].

[10] Smartphone market share. Dosegljivo: http://www.idc.com/promo/smartphone-market-share/os. [Dostopano: 31. 7. 2017].

[11] Somee. Dosegljivo: https://somee.com/. [Dostopano: 5. 5. 2017].

[12] Spring - understanding rest. Dosegljivo: https://spring.io/understanding/REST. [Dostopano: 12. 6. 2017].

[13] Techopedia - android sdk. Dosegljivo: https://www.techopedia.com/definition/4220/android-sdk. [Dostopano: 29. 7. 2017].

[14] The history of java technology. Dosegljivo: http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html. [Dostopano: 8. 8. 2017].

[15] The md5 cryptographic hash function. Dosegljivo: http://www.iusmentis.com/technology/hashfunctions/md5/. [Dostopano: 28. 5. 2017].

[16] Tutorials point - java overview. Dosegljivo: https://www.tutorialspoint.com/java/java_overview.htm. [Dostopano: 8. 8. 2017].

[17] Uradna stran json. Dosegljivo: http://www.json.org/. [Dostopano: 2. 8. 2017].

[18] Virtualni vodič po narodni galeriji. Dosegljivo: http://www.burger.si/MuzejiInGalerije/NarodnaGalerija/Ng_c2.html. [Dostopano: 23. 5. 2017].

[19] Web archive - qr code. Dosegljivo: https://web.archive.org/web/20130129065726/http://www.qrcode.com:80/en/aboutqr.html. [Dostopano: 17. 7. 2017].

[20] What is the difference between android sdk and android studio. Dosegljivo: https://www.quora.com/What-is-the-difference-between-Android-SDK-and-Android-Studio. [Dostopano: 26. 8. 2017].